



Unitec Institute of Technology 3i Research Centre

The 7 Steps of Absolute Information Security with TAIS

First drafted August 2008 by Professor Kay Fielden, David Hughes, Suzanne Sullivan, Alison Clear

Current Version 0.914 (April 2011) maintained by Blueicon Technologies as custodian

blueicon technologies



Table of Contents

Step 1 Secure isolated processes **22**

Step 2 Secure inter-process communication **32**

Step 3 Secure Grid communication **55**

Step 4 Secure process communication **70**

Step 5 Control information accessibility **77**

Step 6 Prevent user access point leakage **94**

Step 7 Calculate the risk function **96**

Introduction

The Theory of Absolute Information Security (TAIS) provides a cookbook formula for both measuring and achieving any desired level of information security in 7 Steps.

When implemented fully, TAIS is designed to:

- Prevent cyber-attacks
- Prevent remote hacking of mobile phones
- Prevent Trojan horses, spyware, viruses or any malware
- Secure classified data and state secrets
- Allow information to self-terminate or be retrieved after it has already been distributed by the person who first received the information
- Prevent the installation of backdoor access by developers that are an enemy of the state
- Enable children to safely share information
- Allow trade negotiators or diplomatic attachés in a foreign country whose computers and passwords are seized to protect their corporate or state secrets
- Allow ministers to communicate with their caucus without fear that the information may be forwarded to the press or passed to members of another party
- Enable corporate leaders to share sensitive information with other organisations on a temporary basis and retrieve documents if the business relationship changes

When TAIS is used in its most advanced form the *TAIS Zeta Environment* (TZE), it allows people to create and share information with any predefined level of security in an open and unbounded grid network.

TAIS is different from other security models because it provides a complete recipe to achieve zero risk security. Other models (at 2011) generally provide good advice on security policy and procedure in order to reduce risk, but never purport to be able to eliminate the information security breach. Other models focus only on system or machine security, but are vague on the topic of protecting information itself. Most worrisome is that a security breach is likely to go undetected, such as in the case of WikiLeaks where authorised personnel forwarded information to unauthorised parties. TAIS specifically prevents information being accessed by unauthorised persons, even if it has been

forwarded to them because information is retained and accessed from the secured grid and is never copied to devices on which it is consumed. Although collectively, the existing security models will indeed help, they are more like an ambulance at the bottom of the cliff instead of extinguishing the source of the fire. TAIS is designed to provide a bullet-proof formula for the protection of state assets at a government or military level.

TAIS provides a clear roadmap for achieving any level of security, and where the implementation costs become too high, it enables engineers to objectively measure the accumulated risk when the deployment is degraded, and most importantly, TAIS allows information access to be governed by this risk function so that critical data, such as the diplomatic content in the WikiLeaks case, can be prevented from being handled by components that do not present a suitable risk profile.

To use the analogy from the aerospace industry, TAIS is like the physical theory behind flight. The amount of lift created from wings, turbulence, air-cabin pressure, landing, takeoff, engines, airstrips, aircraft stability, maximum loading can all be boiled down to applied physics. Just like you can't fly from one country to another by using this theory, you can't use TAIS to secure your information. To actually fly you need a real airplane and the equivalent for TAIS is a software and hardware deployment. In the same way that manufacturers can't build an airplane directly from physics theory, TAIS needs the intermediate step of *design* in order to produce a secure information system. In aeronautics, engineers create detailed designs of aircraft that are based on applied physics, but they also draw from other fields, for example they may design the aircraft so that it is cost-effective to manufacture, or integrate it into other industries so that it can carry heavy loads and use less fuel. Like the Jumbo 747 or Cessna, there are many possible designs¹ for an information system based on TAIS. Finally, information can only be loaded into an implementation from one of these TAIS designs, just like passengers boarding a single aircraft.

This staggered approach results in a complex layered process that will ultimately deliver errors at various stages of production and end use. Conventional (at 2011) information systems contain many hidden errors that are known as *vulnerabilities* that allow *exploits* to invoke security breaches. The problem with *vulnerabilities* is that it is impossible to identify them until it's too late. Worse, a breach may even go undetected. TAIS address the process of going from theory to design to implementation to deployment head on by allowing accreditation agencies to evaluate the

¹ One such design is the BI3 available from www.blueicon.com

accumulated risk of each component. Once the faults are quantified, a simple number called the TAIS Security Risk (TSR) can be derived. TAIS deliberately presents a simple output that can be understood by non-technical people. It is important that a large cross-section of personnel within the organisation can make high level decisions regarding TAIS deployments. These people can use the simple TSR number to compare security products, set IT criteria and determine the kinds of information that can be accessed given a certain level of risk. TSR can be used directly without having to involve IT, who have a different culture and do not understand all the dynamics that comprise the decision in the first place.

TSR is simply the percentage chance that a single information breach will occur over the life-time of a TAIS information system.

Some of the TSR is accumulated during the design and implementation phases in the construction of the product while the remainder is accounted for during the deployment. The theory itself doesn't account for any risk directly because it would be difficult to quantify this risk that the theory itself contained errors. Although a proof is expected for TAIS within 5 - 8 years (2011), it may ultimately be intractable – so it is important that academics and industry agree on the principles for it to be of any objective value.

Methodology

TAIS is comprised of 6 isolated environments that are each secured independently. These are the Alpha, Beta, Gamma, Delta, Epsilon and Zeta environments (and in this advancing order). A requirement for TAIS compliance is that any environment is dependent on the accumulated properties of the preceding environment. However, the more advanced environments are not required to support any specific preceding environment. For example, the Zeta environment factors do not need to be considered when the Epsilon Environment is deployed.

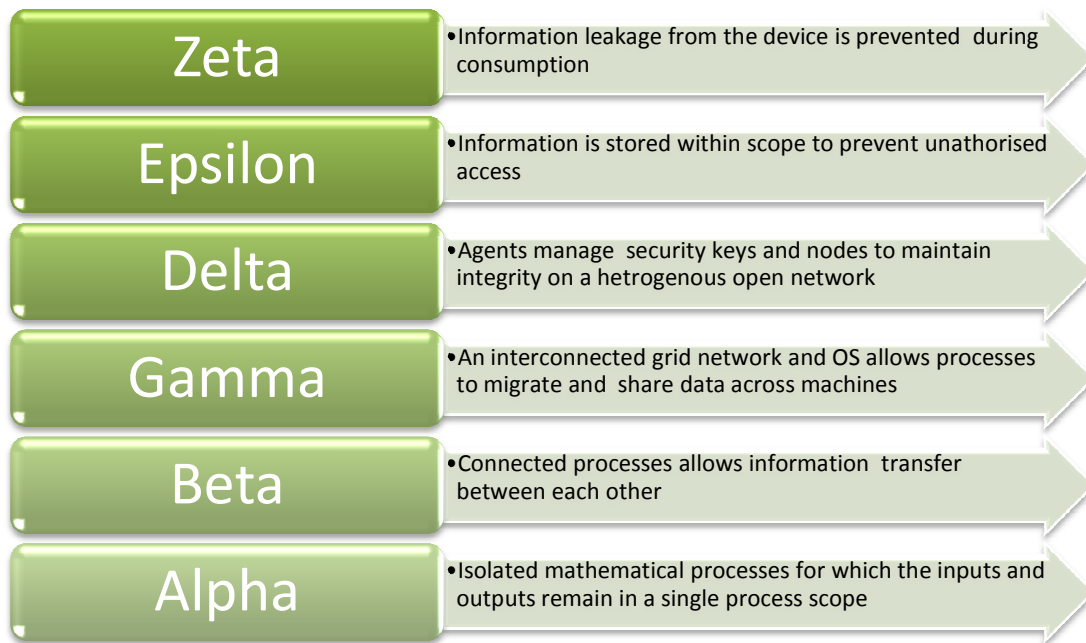


Figure 1 – The TAIS environments can be used in isolation, provided the preceding environments are supported.

Questions and answers

Below is a selection of real-world questions that will give you a taste of the capabilities of TAIS Zeta Environment that is mentioned in the previous Figure 1. Don't be concerned with the detail at this stage; it will become clearer as we progress as many of the answers refer to architecture found at top TAIS layers.

[Q] There are many forms of security today that incorporate user and device access, so how can TAIS be offering anything different?

[A] Today (2011), user and device security is often bundled up in solutions such as Network Access Control (NAC) that will validate a particular device as being a low security risk based on a general algorithm for determining its trustworthiness such as the Operating System, software and patches that have been installed. This is a very crude mechanism because it doesn't take into account necessarily what additional software or documents with execution capabilities (like Macros) have been run since the original installation, and even if it does, it doesn't verify that this code is not a potential Trojan. If it does, then this may make the authorisation too sensitive and a person may be denied access after they review a potentially 'dangerous' document or have installed a software update. There is also very little that this authentication software can do to fully protect against sophisticated attacks where the machine configuration may have been modified and the machine simply misrepresents its true state.

TAIS is different to this because it identifies the machine and the exact path via a sequence of peer-to-peer connections that will link that machine (TAIS is not founded on TCP/IP although it can use it as a transport layer). TAIS does not try to evaluate the integrity of the machine directly, but relies on the person administering the information cluster (called an Infoscope and handled in the TAIS Epsilon Environment) to have verified the integrity of this machine beforehand. This integrity need only be done once (provided the machine hasn't been physically tampered with) because TAIS prevents software that it runs (inside a sandbox) from compromising the machine. This means that the machine will never become compromised, provided all information is accessed via TAIS. In high security scenarios, the machines used to access information may be preconfigured to operate with high (physical) security, monitoring and within a Faraday Cage to avoid Tempest scanners that are looking for electromagnetic radiation leakage.

TAIS is not concerned with the cost implications of how to achieve this high-end security, so for example, it may be cost prohibitive to enforce a lockdown of employee mobiles and laptops so that they only execute the TAIS emulator. However, some organisations may opt for a hybrid approach by accepting some device risk, by preventing the installation of new software.

Another area where user and device access is different from today's systems is in the granularity and scope for which the device access controls have been assigned. Generally today, devices are either recognised as safe or unsafe and are then allowed onto the network. In TAIS, the Infoscope specifies the devices that may be used to access the information, but from the perspective of any single device, the user may still access whatever information is available for which their device has access, so it may log onto the network, regardless of its integrity. This allows less sensitive information to be available. A characteristic feature of TAIS is that compromised machines can safely join the network without compromising other machines.

Another major difference that will become apparent in the TAIS Zeta Environment (TZE) is that each user has access to their own files and these files are dynamic references that are shared between users. This is different from today where users have access to common directories and all share the same files.

There are many subtle differences between the systems in use today (particularly DAC Linux/Windows type systems) and the TAIS model – for example users of TAIS are able to forward documents without the fear that the receiver will not disseminate the information accidentally or deliberately, or looked at another way, a user can be forwarded a document by someone who does not have access to that document.

[Q] There is a big trend today in taking systems offline so that people can work with them on planes for example. The theory appears to enable dynamic collaboration, but what if a person is offline, how can they access their work?

[A] TAIS is not an implementation, it is a theory of securing information and measuring it. It involves an abstract architecture that allows computers to connect adjacently in a peer-peer fashion. As a result, it is ideal for the peer-to-peer connectivity between mobile phones or laptops, such as being conducted by TerraNet in Sweden (2011). TIAS anticipates that connectivity will continue to increase in the mobile area to the point where an individual that is beyond both the range of a base

station and another mobile phone will be extremely rare. Airlines are starting to provide WIFI access and are most probably going to enable connectivity as we move into the future. However, to answer the question fully, it would be true that without connectivity, the user cannot access information in the same way that they couldn't access the Internet without connectivity. On the other hand, the TAIS architecture is already distributed, so it is possible that parts of the database could be separated for extended periods of time and still operate effectively, for example in a military submarine, by locking data at a fine granularity instead of entire files.

[Q] How can an implementation of TAIS really protect schools and children unless it is integrated within the OS so deeply that the user can't use the PC to access anything but the Infoscope system?

[A] This is actually a correct assumption and not just for schools either as it applies to anyone using a TAIS implementation. In any TAIS deployment it is critical that access is limited to the TAIS system and the *execution substrate* (such as Windows) is hidden. TAIS does not try to fix an already broken architecture (Windows), it is merely a recipe for achieving extremely high measurable security. In order for schools to utilise the system, the children cannot have access to PC Operating System directly and must utilise only the TAIS environment (probably via emulated software).

Even in the TAIS Alpha Environment (TAE), it is a major advantage to ultimately remove the *execution substrate* entirely and run TAIS as a dedicated client, because this removes the risk posed by these supporting substrates.

[Q] How can the system protect anyone from say a Macro Virus that is already present on the device that they are using to connect to the TAIS system? If a user is authenticated then what stops them from saving a macro onto the TAIS system that has been put into their spreadsheet knowingly or unknowingly? If the TAIS system prevents users from connecting without approved Anti Virus and Virus definitions, then you are blocking 90% of potential users. Isn't this a problem?

[A] This is a great question because it highlights the difference between the TAIS model and the systems in use today. Although the question is loaded in the sense that it has a number of assumptions, let's break it apart and see how this relates to TAIS.

Firstly, the TAIS system will not protect an end point device from any kind of malware that may already be installed on it. TAIS refers to this as the *execution substrate* and it is a prerequisite (in the TAIS Alpha Environment) that the substrate is not compromised. The mechanism for this is linked to a computing concept called the Malicious Host Problem which states that any layer providing the computation will be able to both read and modify any layer for which it hosts. In fact, the complete removal of this execution substrate is desirable. No doubt this makes it difficult and inconvenient for existing platforms to support both TAIS and their native functionality simultaneously. TAIS is probably more suited to hardware manufacturers that can lock down the machine in some tamper resistant form (like embedded systems). TAIS is not concerned with how this lock down occurs, it is primarily concerned with securing information. Practically, a potential cost effective solution could be to boot into a secure partition that restricts access to everything other than the TAIS system. The mechanisms used to do this all are accounted for in the final risk function. So, for example, TAIS will happily let you run native applications alongside the TAIS emulator, but the risk profile for that device will be high, but this does not affect the risk of the rest of the information system. The information that will carry this additional risk is limited to special information containers called Infoscopes.

The question also refers to authentication and is framed in the Network Access Control (NAC) paradigm. The NAC approach tries to make a best guess validation of each access device. This is good in the sense that it is practical and generally anyone can access the information most of the time, but is bad in the sense that it doesn't make any guarantees about a particular device being secured, and hence the security promise is undermined. TAIS is different because all devices are automatically authenticated and the user is able to access all the information for which their device has clearance. So for example, low security information may be stored in an Infoscope that allows all mobiles and laptops with a particular security profile to be accessed. At the extreme end, Infoscopes that store public information will probably be configured so they can be accessed by all devices, but not all those devices will be able to access information in other Infoscopes.

The next part of this question asks what is stopping a user from copying an infected document from their local machine into the TAIS system. Again, this is a violation of the execution substrate prerequisite of the TAIS Alpha Environment. The user cannot drag anything from their native machine into the TAIS environment without violating Factor TAE 1A (Alpha Environment) and the Alpha Environment certainly doesn't allow for drag and drop from the execution substrate. More to

the point, users can only copy and modify documents that they already have access to within the TAIS system. Even within the TAIS system, there are limitations to where documents can be moved. Information is retained in an outer envelope called an Infoscope and it is impossible to drag information from one Infoscope to another. In this way, non-IT leaders can easily control the dissemination of their information, such as the leader of a political party preventing emails being disseminated beyond their caucus.

This question is leading to another question “If the substrate can’t be used, how can we view conventional software like Word (and Word Macros that may contain viruses)?”. TAIS is a clean-slate approach to security and doesn’t present the Windows Operating System. To achieve this kind of backward compatibility, a PC emulator is required that runs on the TAIS implementation. The desired OS and applications need to be loaded on to the PC Emulation.

[Q] How does simply authorising the user prevent ID Theft in TAIS?

[A] The TAIS system will only prevent ID theft for applications that are hosted within the TAIS system itself. TAIS will not protect an individual from having their identity stolen in a hire purchase type scenario that uses non-TAIS systems. TAIS allows the individual to structure their information in various security levels. Sensitive information such as money or the ability to authorise a transaction, can be kept in secure Infoscopes that can deny access from devices that have inadequately authorised an individual. These Infoscopes act like a virtual safe to keep sensitive information safe while still allowing access to less sensitive data. In practice, an Infoscope may allow virtual devices that represent the accumulation of multiple forms of authentication such as password and confirming a code that is received on the user’s private mobile phone. Depending on the level of authorisation, the virtual device will select a device ID that will unlock Infoscopes with more capabilities for the user. Regardless, if the user doesn’t have sufficient authorisation, then they will be limited to the type of transactions they can commit. At worst, this is impractical, but it does prevent the direct theft of the individual’s ID for applications that are hosted by the system.

[Q] How does TAIS compare to the Bell-LaPadula Model and the Biba Integrity Model?

Bell-LaPadula was designed for military and government applications and is primarily concerned with access control to information. The Biba Integrity Model improved on the design to ensure data integrity by preventing lower authorised users from modifying information created by a higher ranked entity.

TAIS is not limited by this hierarchical view. TAIS' primary advantages are that both information and people do not have to be classed in a ranking order, which is an unnatural fit for information. Quite to the contrary, TAIS specifically does not require information to be grouped in terms of its security profile, but rather information can be grouped in whatever fashion best relates to the natural context to which it belongs. For example, the fluid nature of information causes libraries major categorising issues when books fall into multiple areas. TAIS allows information to not only occur in an unlimited number of contexts by using Set Theory, but it doesn't impose an artificial dominating "security" category that both Bell-LaPadula and Biba do.

TAIS achieves the security with Infoscopes that seamlessly integrate with whatever classification is determined by the creator of the Infoscope, as opposed to Bell-LaPadula and Biba that require the Administrator to do this.

Finally, human error can cause a problem in the Bell-LaPadula and Biba when information or people are incorrectly classified. Furthermore, this problem is exacerbated over time as the deployment landscape shifts and data is not updated. TAIS suffers from none of these issues because information is not centrally classified into cumbersome security structures, so deployment costs and potential breaches are minimised.

TAIS is a holistic solution, so it specifies the implementation requirements to lock down physical hardware as well as prevent information leakage during consumption, where Bell-LaPadula and Biba are quiet on this issue and leave it to the IT engineer.

[Q] How does TAIS compare to mandatory access control (MAC), discretionary access control (DAC) and role-based access control (RBAC)?

[A] All these models focus on access control to information, leaving the question of device integrity unaccounted for.

The security model used by most mainstream operating systems is based on Discretionary Access Control (DAC), which enforces security by ownership. If a user owns a file, he is allowed to set the read, write, and execute permissions for that file. In this model, users control the data at their discretion. The owner of the system does not have total control over the system; the users do.

However, the biggest concern with the Linux model is the danger presented by the root account. This super-user has the power to control all files and processes. If the root account, or a process that runs with its privileges, is compromised, an attacker can take control of the system and its data.

A more secure approach would limit or even eliminate the need for a root account, and shift the power from the user accounts to the owner of the system. This is MAC's approach.

MAC makes the enforcement of security policies mandatory instead of discretionary, as you might imagine from the name Mandatory Access Control. Security policies can be set by the system owner and implemented by a system or security administrator. Once these policies are in place, users cannot override them, even if they have root privileges. With MAC, file and process protection is independent of owners

- Paul Virijevich, Securing Linux with Mandatory Access Controls, 2005

In this light, DAC allows users to own files and pass this ownership to other people. This is fundamentally dangerous, because once a file has been transferred, it is not straightforward to retrieve it. A major disadvantage with this model is that this kind of information breach is likely to go undetected. A breach like this can occur in many ways. For example:

1. Person A is trusted and copies file X
2. Person A copies File X or parts of X to new file Y but saves it with lower security privileges in a way that allows non-trusted person B to access it.

This kind of problem often occurs unintentionally because person A uses software that opens the file; they then copy some of it out into another file. In the process, they may not realise that some of that material may be sensitive.

Both the MAC and TAIS system avoids problems that arise with DAC permissions. For example, let's say that a complex directory structure has a single high security file inside it for which the user can't copy. Then, a DAC system will abort, leaving a mess for you to figure out. With TAIS, the copy is instantaneous because it is done via a single instruction that references the parent directory itself. The copy will always occur because permissions do not pose an issue at the storage level. It's just that the user cannot see the single file for which they are not included in its encapsulating Infoscope. With the MAC approach, this is also not a problem because all the files and users are at the same security level (for that operation).

TAIS is closer to MAC in that there are no owners, but in TAIS users hold their own files and sharing is performed purely referentially. In MAC and DAC, sharing occurs at specific directories. TAIS Infoscopes are an integrated layer that encapsulates these files and governs their transfer between people.

Although the DAC model is possibly more flexible than the MAC based Bell-LaPadula and Biba, it suffers from a similar ailment that people and groups of people must constantly be assigned the files that they can access, leaving the potential for errors. Information must always reflect these rights for every user or group. This is costly and becomes outdated easily. TAIS is simple. Information can be dropped into Infoscopes as needed and otherwise retained in a default outer enveloping Infoscope. For example, if the US government had used Infoscopes during the Iraq war, they could have pulled compromising photos into quarantine Infoscopes before they were disseminated.

The DAC, MAC and RBAC systems pose a major communication inconvenience because they prevent people from sharing information with others who do not have the same security rights. In TAIS, a user can simply drag and drop a folder to another user without concern, because that user will simply not be able to see the information for which they or their access device is not included in the encapsulating Infoscope.

Another area of difference is that DAC, MAC, RBAC have a complex arrangement with each file having read, write or execute permissions. TAIS is much simpler than this because security is viewed purely in terms of accessibility (confidentiality). In TAIS, users can modify any file they have access to and whether they can execute it is irrelevant, because the system prevents software from affecting the host anyway. In TAIS, if you want to achieve the property of non-modification, it's more a configuration issue, not a core security issue. Since users have their own files anyway, a read-only file scenario is achieved in two steps. Firstly, the original file is copied (hence protected) and the copy is shared to various users. Secondly, users access the file with non-editable software such as a PDF viewer. The result is a much clearer security concept – if you have access to it, then it's yours.

[Q] How will TAIS become mainstream if Infoscopes simply prevent information from being accessed when there is insufficient authorisation in cases where the device is not trusted? How does that differ from today where systems already have multiple levels of access?

[A] TAIS is not a business model that desires mass adoption. Rather, it is purely concerned with protecting information and being able to measure the resulting deployment. From a business perspective, it is likely that TAIS will find niche opportunities in military and government and gradually extend its reach as virtualised Operating Systems provide the inter-operability that ordinary users would expect. TAIS differs from systems today that have multiple levels of access because it allows the user to be authorised at various levels and this level of authorisation gives them the capability of viewing certain levels of information. In TAIS, if a user is authorised on a low integrity device, they will still be able to move and copy highly sensitive information – they just won't be able to see those sensitive files and folders.

[Q] It is generally understood that if something can be displayed on the screen then it can be copied, hence the problem the music and movie industries are facing. How could TAIS prevent this?

[A] The statement in the question is absolutely correct and TAIS is no exception. What TAIS does do, is prevent access to sensitive information to a device that may have monitoring software installed in it. The reason the music industry would probably find little benefit from TAIS is that they have no control over the devices that the customer utilises. Many industries such as banking also have limited capability of controlling the user device, so TAIS would also present limited use in these circumstances. However, a banking device that was tamperproof and distributed to customers would work well with TAIS.

[Q] How does TAIS differ from Common Criteria?

[A] Common Criteria is an accreditation programme that verifies that a system will always operate in a secure state. It is an expensive process and can cost up to 1 million USD to complete. There are a number of issues with Common Criteria.

1. The time and effort to evaluate a product is so long that it may become obsolete before release.
2. Updates become an issue because the system state undergoes change and must be re-evaluated
3. Common Criteria is extremely broad and uses a “Target of Evaluation” to cater for a diverse range of systems.

TAIS is different to Common Criteria and is not prey to the same pitfalls. In (1) and (2) above, accreditation is not required at the application level since the Malicious Client Problem is resolved and the TAIS system itself is a security framework. However, the TAIS emulator and execution substrate should undergo some form of accreditation. In TAIS, software loaded onto the system at any point will not affect the integrity of the deployment.

In (3) above, Common Criteria is not specific about Information Security, it deals more about System Security. TAIS targets the problem of Information Security entirely. To use an analogy, Common Criteria would be like checking an aircraft structure before the flight, but a perfectly functional aircraft doesn't always guarantee a safe passage. In this analogy, TAIS would ensure that the pilots, weather and airport security were checked too.

[Q] How does TAIS differ from NAC?

Network Access Control (NAC) attempts to unify endpoint security by making an assessment of integrity based on Operating System, Software, Patches and machine history. This is important because once a computer is allowed to join the network, they breach the firewall and automatically open a number of potential attack vectors. For example, they could connect to other computers on the same LAN that would otherwise have been off limits due to the firewall.

NAC is not an information security model, but a practical unit of armoury in the same way a bullet proof vest is not a police force.

[Q] How does TAIS avoid the classic problem of moving information from one security level to another, as presented by the Multi-Level Security or Bell-La Padula models, for example when a Top Secret file is “Sanitised” and moved to a lower level.

[A] By using Infoscopes to encapsulate information usage, different levels of security can be represented by storing information in different Infoscopes. TAIS must go through a similar process to the Bell-La Padula model, whereby the system is overridden in order to make the shift from one security level to another. In TAIS this is formalised where the owners of both the source and destination Infoscopes must agree on the information being transferred. Although this is only slightly stronger than the Bell-La Padula model, provided that the destination Infoscope prevents information leakage by specifying appropriate device access, then the information can still be locked down. At some point in the future, if the information is understood to have been leaked into a less secure zone, it can still be retrieved and pulled into the secure Infoscope.

This is a challenging problem for any security system. Consider de-sanitising a Word file that may contain hidden “Undo” data from previous edits. The data essentially must be reconstructed in a labour intensive process. A simple *copy* and *paste* could carry some of this sensitive hidden data into the lower security level.

[Q] How will TAIS become established if users must have adopted the system in order to communicate? This is the same problem of how the first person who invented the FAX would have faced when nobody had one, how does TAIS get off the ground?

[A] TAIS does not address the issue of critical mass of adoption, simply because the property of pervasiveness is not a security factor. However, it is likely that niche products such as a messaging system that allows executives from two companies to overlay a secure network across their existing independent organisation infrastructures could be useful. Such a product could be easily collapsed when it is no longer required without having to involve the two IT departments.

[Q] If TAIS aims to remove the *execution substrate* then every device must operate on dedicated hardware and have biometric readers in order to obtain high-end security. Isn't this too expensive? And in the case where the *execution substrate* is used, what would stop key logging software?

[A] TAIS is not a theory that is designed to fix already insecure architectures, so it cannot simply be applied to systems like Windows, Linux or even the Web in order to make them secure. Instead, TAIS must be deployed with specific rules in order for it to be valid. In TAIS, the higher the desired security, the more expensive the deployment as costs to prevent machine tampering, information leakage and biometric authorisation increase.

[Q] How can TAIS result in a secure implementation when software will always contain undetected faults that could be used to launch an attack by hackers to break into the system. How does TAIS address these practical realities of software development?

- Question by Shen Hongkang

[A] A great question. Lets identify the vulnerabilities that can occur. The software written to run on TAIS systems cannot induce a vulnerability because that client software is pre-scanned to ensure that it has zero dependencies (no library calls) when implemented in software (lets consider the software case here only because this is your question). This inability to affect other TAIS processes, the TAIS Emulator or the substrate (if it exists) is a solution to the Malicious Client Problem. Zero dependencies would otherwise make the software fully introspective and effectively useless, but a 'safe' communication method required by TAIS called the Reference-Membrane and Context-Reduction-Engine enables communication without vulnerabilities. However, preventing vulnerabilities from residing in the TAIS Emulator itself (as opposed to software running on the TAIS Emulator) is another problem altogether. TAIS addresses this in 2 ways. (1) the architecture is relatively simple when compared to modern day Operating Systems and so the number of lines of code is much less and analysis much easier, and (2) there is no direct network access to the substrate or TAIS emulator itself, so direct remote attacks are not possible. Other than this, TAIS requires that the machine must adhere to the specification and any deviation will accumulate Design-Time Factor risk. Since there is a much smaller attack surface area (which doesn't change as new programs are loaded) the integrity of the TAIS Emulators can be proved secure or the risk assessed during accreditation.

[Q] How can TAIS be applied in practice? Is it like anti-virus? How could this technology be used by IT today?

- Question by Shen Hongkang

[A] Since TAIS is a general theory of security, it cannot be applied directly in an information system. There are a number of intermediate stages that need to occur. Firstly, a design specification must be drawn up (such as the BI3). From this design, software can be constructed or a hardware circuit manufactured such as in Field Programmable Gate Arrays (FPGA fabric). If it is a software solution, then it will need to execute on some supporting substrate such as Windows or Linux. During this process all the abstract design criteria must be resolved (covered later, also review the BI3 specification for how this is done). Resolving criteria of abstraction means that before BI3 Emulator can operate, everything unspecified or abstract needs to be ultimately defined exactly. Resolution is a complex process, but the BI3 specification and bluebrick™ Java implementation is an experimental research deployment that demonstrates the resolution process. There are no known commercial implementations ready for securing IT infrastructure (as at 2011), but there may well be within a few years.

To answer this question in another way, the TAIS approach is a clean-slate design in that it foregoes backward compatibility for security. This means that information must be transferred onto the TAIS deployment and won't run directly. Windows systems will need to execute through emulation of a PC that runs on the TAIS deployment. It is not like anti-virus in any way because it does not aim to correct faults on flawed computing architectures. Rather, it aims to prevent fault altogether by using an alternative approach.

[Q] Manufacturing companies that develop the hardware on which TAIS will be loaded are similar to how Windows comes pre-installed on generic PCs and laptops. In this comparison, manufacturers of hardware specific to Windows have a clear idea of the software that will be loaded onto their machines and so can design the hardware accordingly but if the TAIS specs are too vague then it will be problematic for manufacturers to produce hardware for its use; so therefore developers would have to build the TAIS implementation around the hardware. Am I on the right track?

-Nick Schofield

[A] This a great question because it highlights the nature of TAIS as a theory. You are absolutely right in your question, hardware manufacturers could not produce (meaningful) hardware based on TAIS alone. We need a specification which resolves the criteria of abstraction such as the BI3. The abstract concepts found in TAIS are fully undefined, for example, TAIS Alpha 1a states *“The execution substrate of the TAIS Alpha Environment is not compromised”*. Clearly, this isn’t enough for manufacturers. Therefore a specification must clear this up. Some specifications like the BI3 still have some abstract components that must be resolved during implementation – but the range of architectural possibilities is contained to enable interoperation regardless of the implementation choice. The TAIS model really highlights how this is a continuous scale from abstract theory to concrete specification and there are a great variety of positions that an architectural description can take along that scale. This is very common in the Object Orientated methodology, which allows total freedom of abstraction, provided all abstract methods are implemented prior to linking. The reality is that in both hardware and software, before anything can exist and execute (operate), somewhere the exact manifestation will need to emerge.

[Q] How can it be possible to prevent a remote party from engaging a passive attack that takes advantage of flaws or backdoors in software simply by using a Context-Reduction-Engine and Reference-Membrane? Surely, there must be some scenario for which this defence configuration won't hold true and an attack can be made? For example, let's say that a military contractor is actually a double agent working for the enemy and has developed code that activates a backdoor when a certain condition is met – let's say when the code processes the user name "*John Andrew-Jackson Smiley*"?

[A] The TAIS Beta Environment (TBE) locks down the capabilities of the military contractor's code as follows:

1. It cannot make any library calls
2. It cannot reference anything outside of itself
3. It must communicate with a minimal set of narrow instructions

The Context-Reduction-Engine (CRE) is more secure when there are fewer Context-Handlers and each is narrow and safe in scope. In an ideal implementation, the contractor's code resides in a TBE process that has no knowledge of any previous instruction. Its capabilities are limited to the list of Context-Handlers available in the CRE and the developer's TBE process can't make a connection to another computer to disseminate information.

To reduce the TSR, the number of Context-Handlers must be reduced (5@B3b) and the scope of the Context-Handlers themselves reduced (2@B3b). A low risk for non-trusted code can be achieved by outsourcing a series of a modular functions that each operate independently with customised CREs for their specific requirements. A high TSR could occur in an implementation that built a broadly functional application that accessed multiple generic CREs with many Context-Handlers that were not required by the process.

Run-Time VS Design-Time

Throughout the explanation, the resulting security recipe is an accumulation of the environmental factors that are tracked in two groups:

TAIS Security Factors

Run-Time Factors	Design-Time Factors
Run-Time factors need to be considered for each deployment as they constitute variables that are found in the environment	Design-Time Factors need be considered only once, their properties are inherited by the deployment as they emerge from the design

The Design-Time Factors are locked down progressively over the 7 steps within TAIS, resulting in the design specification and implementation.

All the factors will combine to form a measurable risk function (Step 7), but the Design-Time Factors should be addressed by the manufacturer of the TAIS implementation, leaving IT to address the Run-Time Factors.

If you're still unsure of the difference between the run-time and design-time factors, don't worry. Take a look at the simplest environment, the TAIS Alpha Environment and understand the TAIS Alpha Factors. Next, see which of these factors fall under design-time and which are run-time. Then, try to understand why design-time factors can be eliminated with accreditation.

Step 1

Secure isolated processes

ALPHA ENVIRONMENT

The TAIS Alpha Environment (TAE) is the simplest of the 6 TAIS environments and it is barely useful from a practical standpoint as there is no IO communication with either people or other processes. It could, for example be used to calculate pi or calculate prime number factors for asymmetric-encryption keys such as in the RSA algorithm. Perhaps the TAE is most useful because it allows us to understand and study a small piece of TAIS before it explodes in complexity.

Summary of TAE Criteria of Abstraction

1. The execution substrate of the TAIS Alpha Environment (TAE) has these properties:
 - a. is not compromised
 - b. has no network access
 - c. remains powered on and has no physical failure
 2. The TAIS Alpha Environment has the following properties:
 - a. TAIS Alpha Environment boots and hosts isolated computationally complete single threaded processes
 - b. TAE processes cannot share references or affect each other
 - c. TAE processes cannot compromise² the TAIS Alpha Environment or the execution substrate
 - d. TAE processes have no boot parameters
 - e. TAE processes must contain zero dependencies³
 3. Secured information must be managed from within a TAE process
-

² Such that the Malicious Client Problem is solved

³ No library or external calls allowed (such as to the Operating System or hardware)

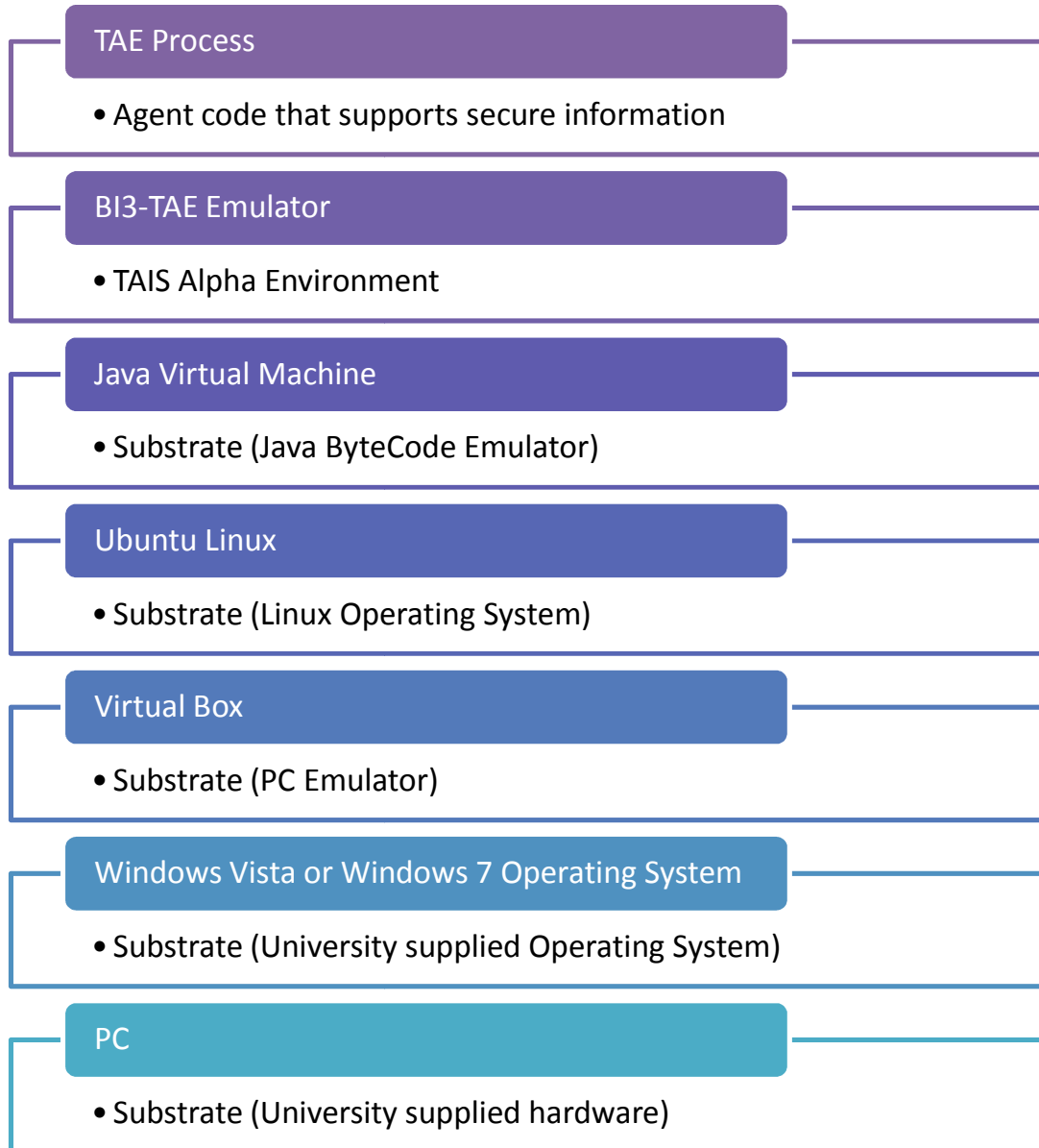


Figure 2(a) – There are many possible substrate configuration possibilities, above is the stack used during the BI3 Ethical Hacking Challenge 2011.

The TAE comes into existence when an implementation is deployed that adheres to each of the TAE Factors. Any deviation will introduce an information security risk and can be calculated in Step 7.

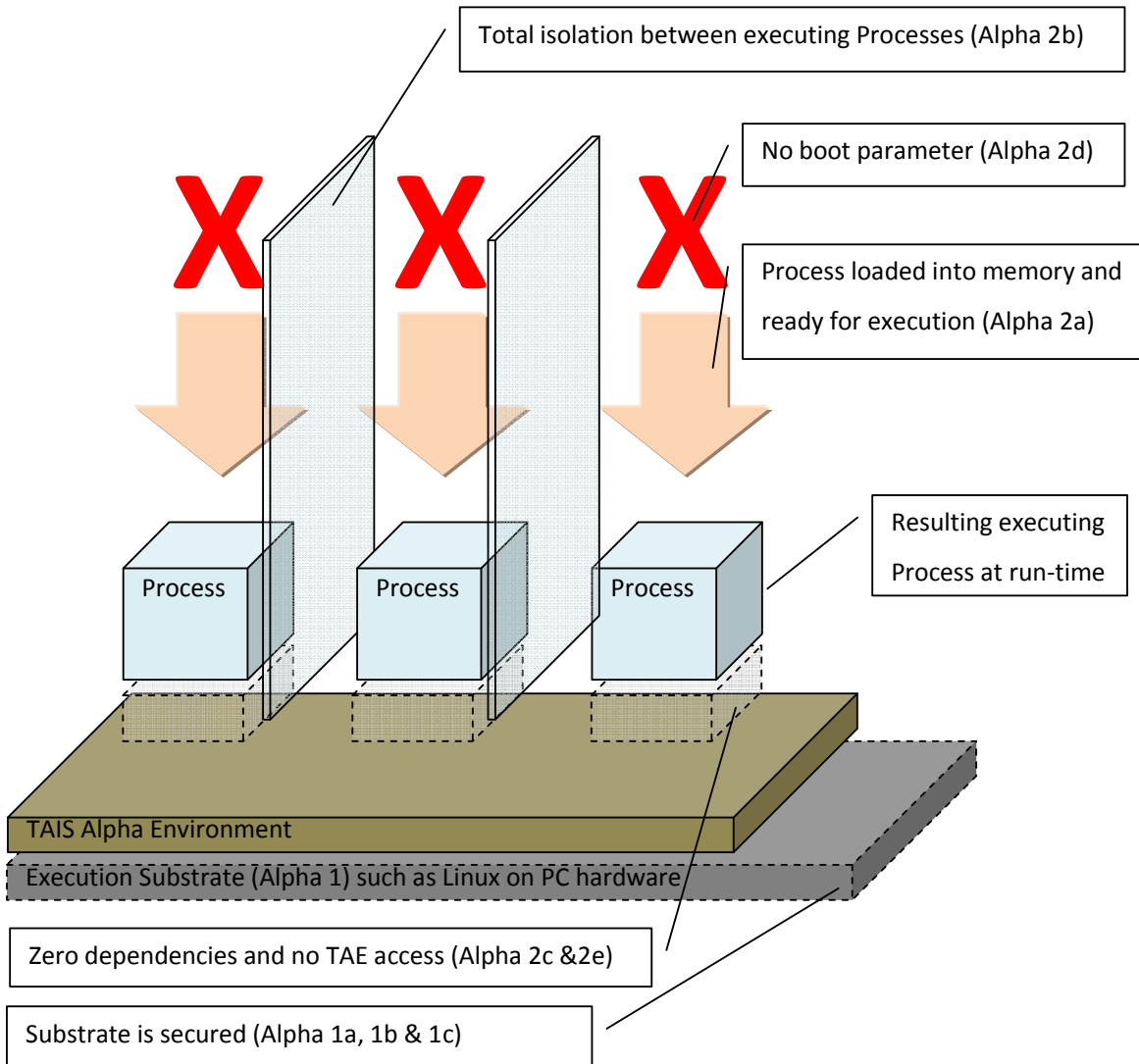


Figure 2(b) – Every Process on the TAIS Alpha Environment is secured, provided the TAIS Alpha Factors are true – the relevant factors are listed above.

Alpha 1a

The execution substrate of the TAIS Alpha Environment is not compromised

The most likely deployment scenario (at 2011) for the TAE is to operate as an emulated software environment that resides on some cost-effective computation substrate. This factor can be eliminated entirely if the TAE is comprised directly in hardware, such as a Field Programmable Gate Array (FPGA) fabric or secured embedded systems.

Criteria of abstraction

- 1@A1a The execution substrate must start its life being uncompromised
- 2@A1a No additional software other than the TAE emulator may be installed onto the execution substrate
- 3@A1a The execution substrate may not be modified or tampered with
- 4@A1a The execution substrate must restart to the same state every time

The last point above has implications for portable devices like mobiles and laptops that cannot always guarantee that the device will be physically secured.

Alpha 1b

The execution substrate of the TAIS Alpha Environment has no network access

The substrate must be protected from remote network attack by eliminating all network access.

Criteria of abstraction

- 1@A1b The execution substrate must remain isolated from other computers

Alpha 1c

The execution substrate of the TAIS Alpha Environment remains powered on and has no physical failure

Factor 1c is essentially the notion of availability and there is a weak argument that this should not be considered as a security element. A strong case for including physical failure as a security factor is that a fault could cause people to use an alternative and possibly non-compliant TAE processes to compensate for the loss of this otherwise secured TAE.

Criteria of abstraction

1@A1c The execution substrate remains available throughout its use

Alpha 2a

TAIS Alpha Environment boots and hosts isolated computationally complete processes

Factor 2a is extensive as it stipulates a fully functional computer with CPU, memory, branching statements and useful numerical primitives.

Criteria of abstraction

- | | |
|-------|--|
| 1@A2a | Any computationally complete architecture may be used including von Neumann logic or exotic quantum processing logic |
| 2@A2a | The architecture must support multiple processes (that can only access their own reserved memory, see also 1@A2b) |
| 3@A2a | Each process has a predefined number of instructions that will be processed each second and the processing rate remains unchanged regardless of the state of any other process being hosted by the TAE |
| 4@A2a | A process starts as a single thread ⁴ and remains as such until the program finishes as there is no instruction to establish a new thread or execute new code. |
| 5@A2a | All state change occurs on a per instruction basis and directed to a Random Access Memory (TAE-RAM) of an arbitrary but consistent bit length |
| 6@A2a | Instructions can perform basic math in memory |
| 7@A2a | The manner in which the process code is loaded or stored remains undefined |

⁴ Multi-threaded applications must utilise their own software for virtualised thread swapping

Alpha 2b

TAE processes cannot share references or affect each other

Factor 2b ensures processes are fully isolated.

Criteria of abstraction

1@A2b Each process is allocated run-time memory and process instructions may only reference this memory

Alpha 2c

TAE processes cannot compromise the TAIS Alpha Environment or the execution substrate

Factor 2c is a general provision that ensures the Malicious Client Problem is eliminated⁵.

Criteria of abstraction

1@A2c There should be no state that the TAE process could possess which would otherwise adversely affect the TAE or the execution substrate

⁵ The factor is linked to 2e which covers the case if a substrate is used, whereas 2c covers the whole principle. For example, a software implementation must scan for library calls made in the TAE-Process (this is 2e) but if the whole TAE Emulator resides on a platform like the Java Virtual Machine, then 2c covers this regardless down through the stack. In this example, 2c would mean that it should not be possible to create an exploit using the Java "Hot Spot" compiler that could attack the host even though there were no library calls.

Alpha 2d

TAE processes have no boot parameters

Factor 2d is deliberately limiting and is extended in the TAIS Beta Environment (Step 2 next).

Criteria of abstraction

1@A2d All processes should boot directly from their static initial state in such a way so that they produce the same result every time

Alpha 2e

TAE processes must contain zero dependencies

Factor 2e refers specifically to emulated software that enables TAE processes to execute with the same code format and substrate loading sequence as the emulator itself. Factor 2e is the reason why emulation can ultimately be tamed to prevent the Malicious Client Problem.

Criteria of abstraction

1@A2e No instruction may exist that permits the invocation of any software external to the process, in particular linked external library calls served by the TAIS Alpha Environment or execution substrate

Alpha 3

Secured information must be managed from within a TAE process

Practically it is unlikely that secure information will actually be stored on such a simple environment as the TAE, however, if required then the TAE process must store the secure information within its own memory structure. No information must be stored on the substrate.

Criteria of abstraction

- 1@A3 All information requiring protection should be stored within the TAE process.
- 2@A3 Access to the protected information must be performed directly from TAE-RAM and any mechanism to do so must either present exactly zero additional risk, or whose additional risk must be accounted for in the TSR

TAE Criteria of Abstraction

Run-Time Factors	Design-Time Factors
Alpha 1a, Alpha 1b & Alpha 1c, Alpha 3	Alpha 2a, Alpha 2b, Alpha 2c, Alpha 2d, Alpha 2e

To understand the implications and differences of the run-time and design-time factors, let’s look closely at the TAE Factors above. The design-time factors should be able to be addressed at the design and implementation stages. The run-time factors make up the remainder of the deployment and are not included as part of the architecture.

In practice, many machines that could present the substrate layer for TAE and could become compromised if they are not physically protected at all times. Therefore in a recommended variant, the substrate can be removed entirely, if the TAIS Alpha Environment implemented directly from hardware⁶ - meaning security⁷ is guaranteed provided the hardware is tamper-proof.

⁶ Manabars Technologies Ltd (MTL) achieved two patents for such a machine, *Computational Device For the Management of Sets, 2007* and *An Abstract Platform to Facilitate the Interoperability of Information, 2007*

⁷ Security of access doesn’t include an unexpected shutdown if the device is fails.

Step 2

Secure inter-process communication

BETA ENVIRONMENT

The TAIS Beta Environment (TBE) extends the usefulness of the TAIS Alpha Environment (TAE) by introducing additional primitives that enables secure communications between processes on the same TBE-Emulator.

Secure inter-process communication on the TBE is established with:

1. The *Reference-Membrane* initially pioneered⁸ by Professor Fielden
2. The Context-Reduction-Engine (CRE) that enables safe remote access between processes

DEFINITIONS

Reference-Membrane: A data transfer link that ensures all memory references are copied and do not refer back to the source process.

Context-Reduction-Engine: Software executing on a TBE process that manages all the communication with another process in a manner which prevents the remote process from disrupting the behaviour of the local process beyond its designed operational boundaries.

⁸ See An Interoperable Information Infrastructure Model (III-Model), <http://blueicon.com/downloads.html>

TAE is extended into the TBE as follows:

- **Booting**

Each TBE process is booted with a *Set* parameter

- **Memory**

The TBE introduces 2 new types of memory:

- Referential Memory
- Static Memory

- **Primitives**

6 additional primitives enable the TBE process to access different data structures:

- Execution
- Float
- Whole
- SOWA
- Duplex
- Set

Summary of TBE Criteria of Abstraction

1. The TAIS Beta Environment inherits the TAIS Alpha Environment
2. The TAIS Beta Environment process
 - a. must boot with a single Set parameter
 - b. has access to
 - i. the SOWA primitive
 - ii. the Set primitive
 - iii. Float, Whole and Binary primitives
 - iv. the Duplex primitive
3. All communication between TAIS Beta Environment processes must pass through both a
 - a. a Reference-Membrane and
 - b. a Context-Reduction-Engine

(Note TBE numbering has changed since version 0.912)

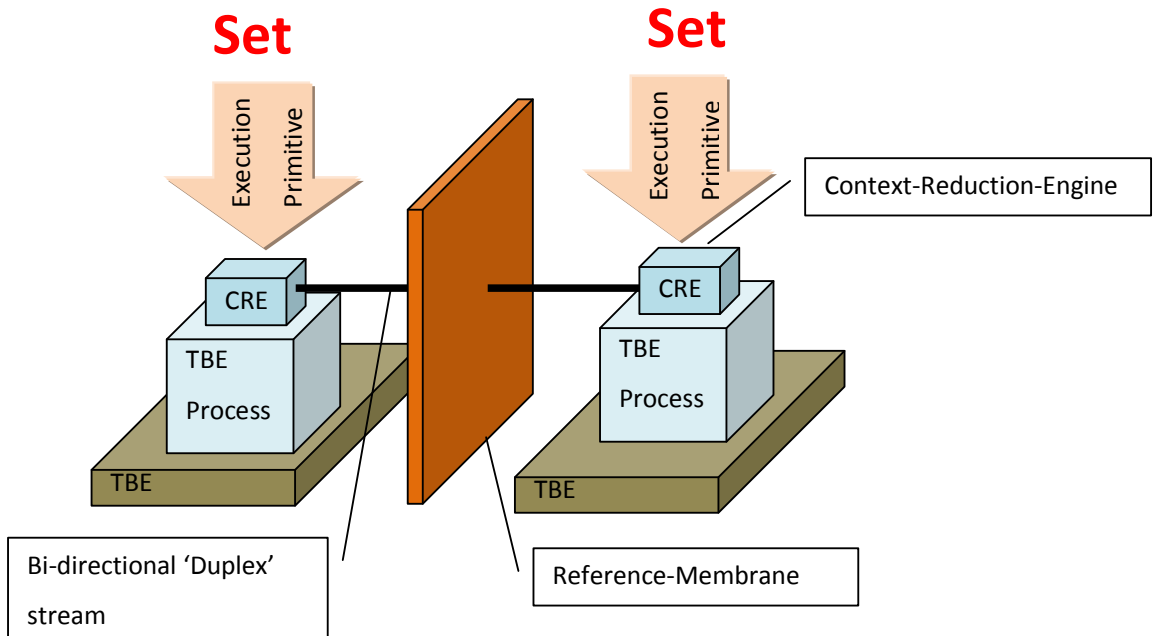


Figure 3a – Secure inter-process (and inter-machine) communication is available on the TAIS Beta Environment (TBE) by using a Duplex type, a Context-Reduction-Engine (CRE) and a Reference-Membrane.

Beta 1

The TAIS Beta Environment inherits the TAIS Alpha Environment

Where otherwise stated, the TBE includes all the functionality defined in the TBA

Criteria of abstraction

- 1@B1 The TAE-RAM from the TAE is replaced by the TBE-RAM in the TBE and is controlled entirely by the Execution primitive of the TBE process
- 2@B1 The TAE process from the TAE is replaced by the TBE process in the TBE and the image loading is now resolved (reference 7@A2a) as the Execution primitive is a Referential-Memory store

Primitives

The TBE primitives are the collection of types for which all information on the TBE must ultimately be encoded in. These primitives work in parallel with the TBE-RAM. The TBE primitives create additional data storage, access and transmission framework that allows information to be loaded on and off the TBE primitives directly from within the code without any library calls to the TBE or TBE substrate. In the design specification, these primitives may be mapped to instructions that fetch or manipulate data residing on another microprocessor.

Primitives in the TAIS Beta Environment

Primitive	Description
Execution	Determines the code segment of a TBE process and is not accessible introspectively
Float	N-Bit floating point number
Whole	N-Bit integer number
SOWA	Static Once Write Array (SOWA) is an N-Bit array that cannot be modified once established
Set	A referential Set structure that can be assigned each class of primitive only once at a time
Duplex	Transfers primitives bi-directionally between TBE processes

The Set primitive

The Set primitive acts as a unifying force between all the other primitives:

- The Set provides control of access by encapsulating other Sets
- The Set can store exactly 1 reference to any other primitive as well as any number of children Sets ordered from left to right (references may occur any number of times)
- The Float and Whole are permanently assign and default to zero

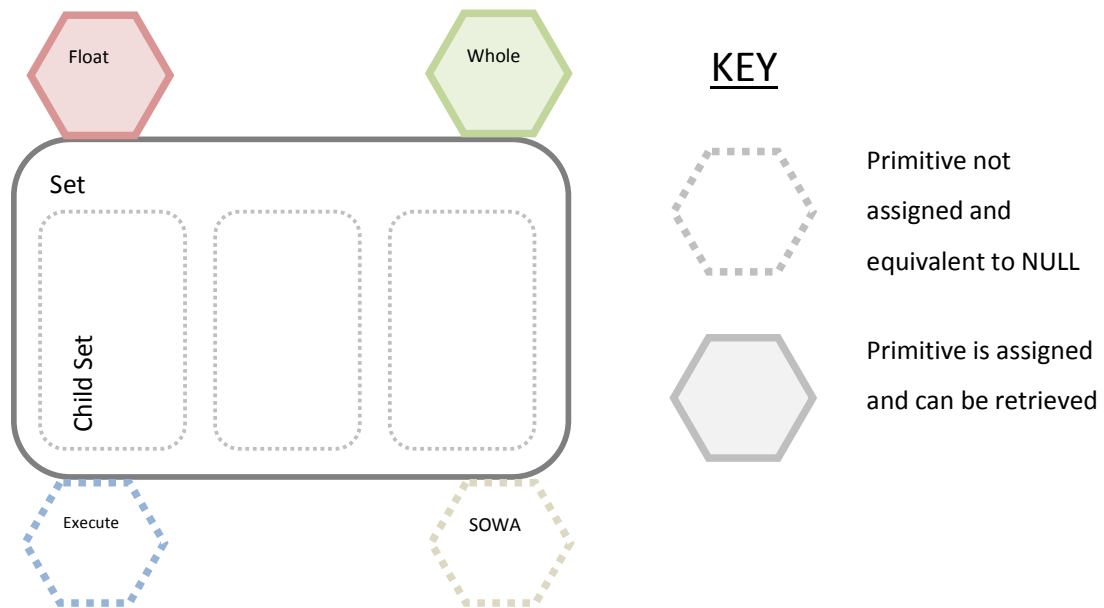


Figure 3b – The Set primitive can store every other primitive except the Duplex since it is transient.

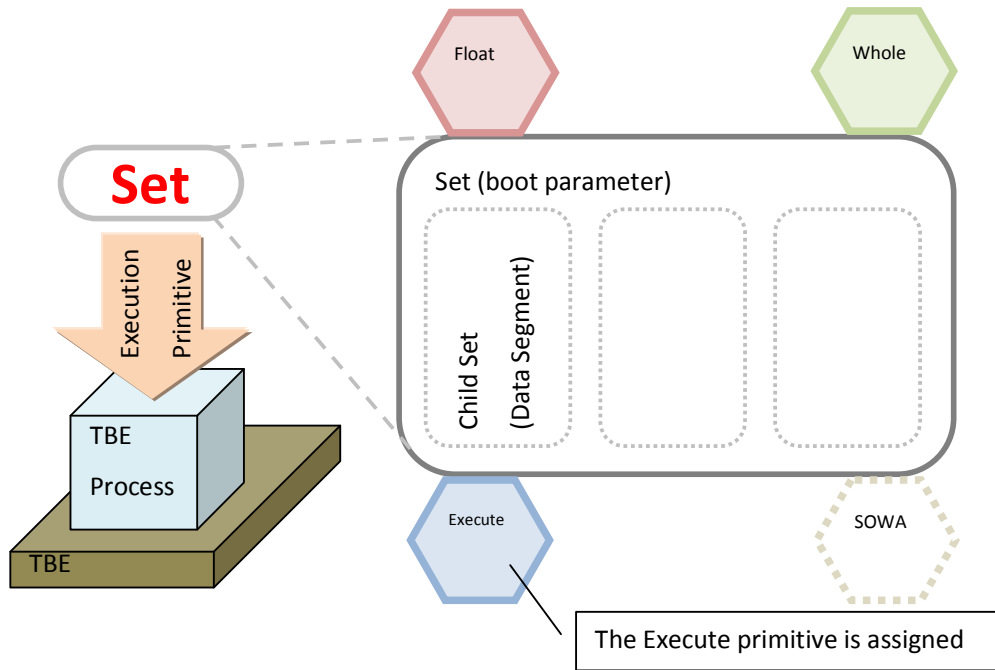
Beta 2a

The TAIS Beta Environment process must boot with a single Set parameter

The initial conditions (starting state) of the TAIS Alpha Environment (TAE) process are preset so that it executes identically on each run. In contrast, the TAIS Beta Environment (TBE) allows the same TBE process code to be run with different parameters each time⁹. Some of these parameters enable sharing of information between processes on the same machine and between processes on different machines.

Criteria of abstraction

- 1@2a The TBE must have access to a Boot Set whose structure can only be altered by the TBE process itself.
- 2@2a The TBE Boot Set must have its Execute primitive assigned with the code segment of the TBE process itself.



⁹ In comparison to the TAE, the TBE Boot Set parameter is only one driver for the TBE execution state. Since the TBE process has access to the SOWA permanent storage, it is able change its state by accessing data generated on previous executions.

Memory

There are two additional memory structures presented by the TBE that go beyond TAE-RAM of the TAIS Alpha Environment.

D E F I N I T I O N S

Static-Memory

Static-Memory includes the Execution and SOWA primitives. This data content will never change once created and may be accessed by multiple processes concurrently without either processes being able to affect each other.

Referential-Memory

Referential-Memory includes the Set primitive as well as any local TBE-RAM references created at run-time. The references can store dynamic links to other Referential-Memory or Static-Memory and cannot be shared by different processes.

Beta 2b-1

The TAIS Beta Environment process has access to the SOWA primitive
--

The SOWA primitive has the primary property that it can never change once written. Static-Memory is designed for efficiency of transmission and storage because it can be cached and shared between processes safely. Static-Memory can be cached, either on chip to resolve the Von Neumann Bottleneck problem¹⁰ or on a grid to help solve locking and latency issues.

Static-Memory is accessed with the SOWA primitive, which can be used to retrieve its data image.

Criteria of abstraction

- | | |
|---------|--|
| 1@B2b-1 | Each SOWA block written must be keyed to indicate if it is encoded as non-typed binary data, typed Float ¹¹ or typed Whole ¹² data |
| 2@B2b-1 | The TBE must contain at least one SOWA Static-Memory form where each additional SOWA form must present a different storage characteristic |

¹⁰ The Von Neumann Bottleneck acts as a glass ceiling (at 2011) for further advances in hardware performance gains because the bus that moves memory onto the chip is unable to keep pace with the demand for memory as the number of cores available on the chip increases. By keeping memory segments static, chunks of memory can be cached without fear that they may be altered by another process, making this approach ideal for Field Programmable Gate Arrays (FPGA).

¹¹ Floating point of N bits

¹² Integer of N bits

Beta 2b-2

The TBE process has access to the Set primitive

The Set primitive is designed for encapsulating data with scoped access and is founded on the principles of Set Theory. The Set primitive is flexible and powerful, for example, a Set can be configured to remove itself from any number of lists with only one instruction required for each list removal.

Criteria of abstraction

- 1@B2b-2 The TBE process can create and modify a Set primitive that it has access to
- 2@B2b-2 The Set primitive allows access to any number of child Sets that it contains but does not provide access to any parent Set containing the Set for which the TBE process has access
- 3@B2b-2 Child Sets can reference any other Set provided it is not already referenced by another TBE process and must have a consistent ordering
- 4@B2b-2 Each Set can be assigned an Execute primitive
- 5@B2b-2 Each Set can be assigned exactly 1 primitive for each type of SOWA specified in the design specification
- 6@B2b-2 Each Set is permanently assigned a Float and Whole primitive whose default values are null

Abstract Memory Layout

Although the design specification (such as the BI3) will ultimately determine the layout of the microprocessors, TAIS handles a conceptual representation of the memory, CPU and substrate layout.

The figure on the next page shows how Referential-Memory can access shared Static-Memory and yet still preserve isolation of TBE processes. In the TBE-RAM, this could be by way of assignment as directed by instructions in the code segment represented by the Execution primitive. For example:

```
private sample(SOWA s)
{
    private Set localSet ;
    localSet.assignSOWA(s) ;
}
```

The above code, built for a fictitious design specification of TAIS is moving a Static-Memory reference between the two forms of Referential-Memory.

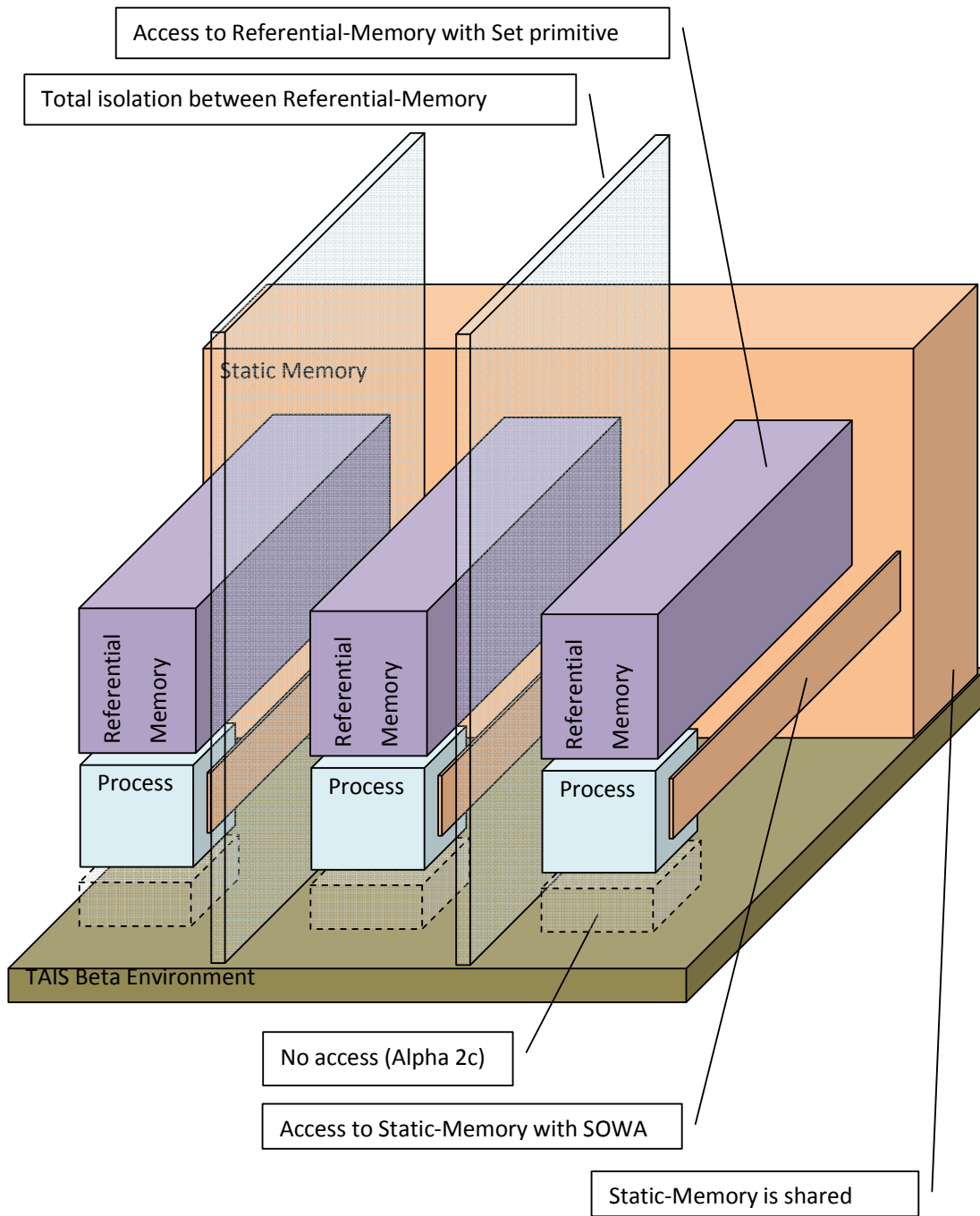


Figure 4, Referential-Memory and Static-Memory both preserve isolation between processes

Linking Referential-Memory and Static-Memory

Data can be accessed with both Referential-Memory and Static-Memory by using the Set primitive.

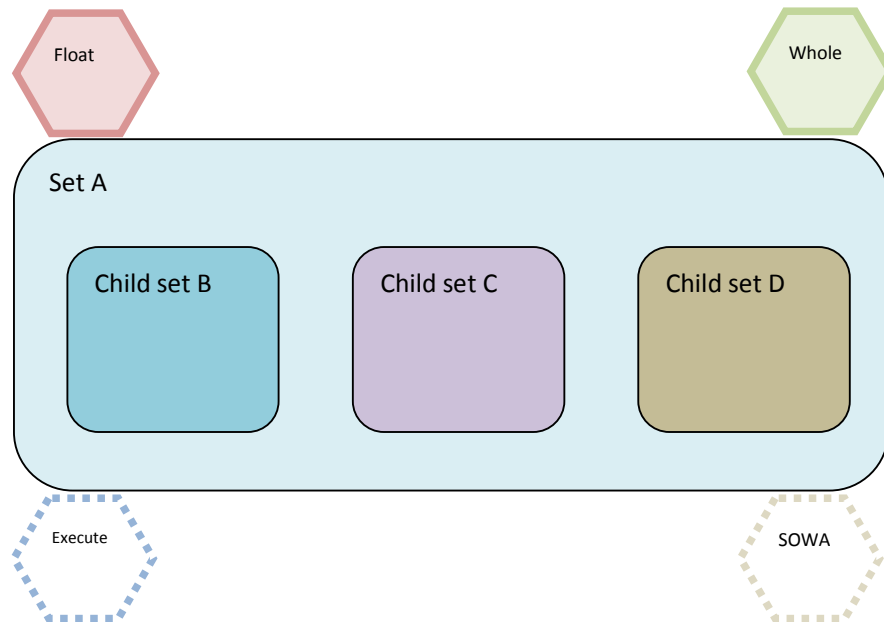


Figure 5, an ordinary Set structure

The child Sets B, C and D above are all Referential-Memory. Sets can contain any other Set reference, any number of times but have a consistent ordering from left to right.

Static-Memory is shared

Static-Memory is attached to the Set as a SOWA or Execution primitive. For example, both Set A and Set B could have attached a SOWA that represents U2's "It's a beautiful day" in an MP3 format.

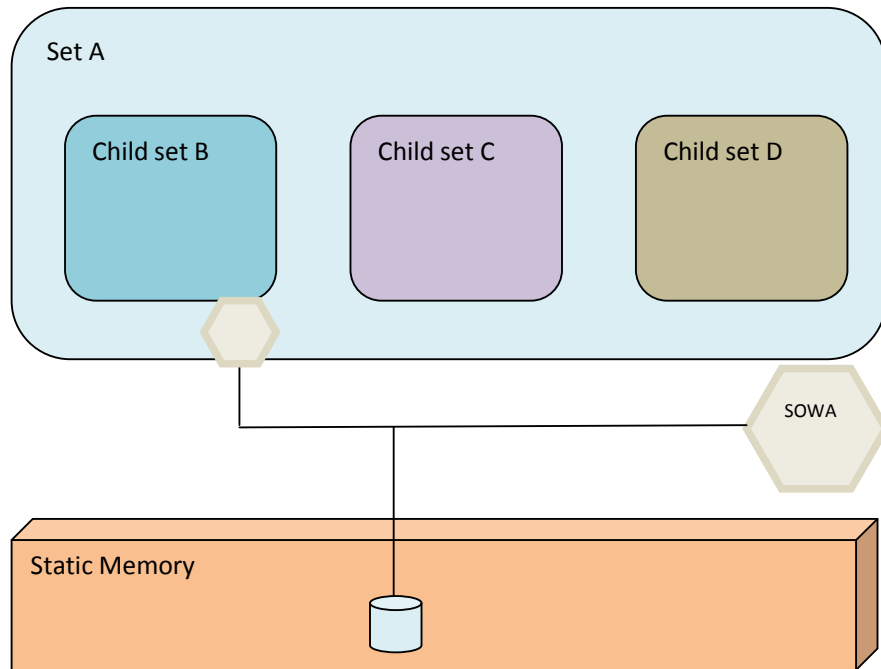


Figure 6a, a single SOWA is shared by multiple references

Beta 2b-3

The TAIS Beta Environment process has access to Float, Whole and Binary primitives

The Float, Whole and Binary primitives are used to store and access data blocks which are each the same size and matches the number of bits for which the TBE is configured (for example 64 or 128 bit TBE machine size).

Criteria of abstraction

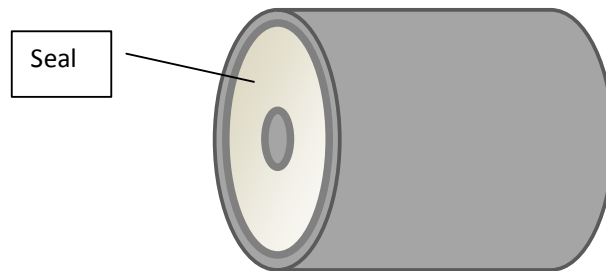
- 1@B2b-3 The number of bits used for the Float, Whole and Binary primitives cannot change once implemented

- 2@B2b-3 The bit encoding of the Float, Whole and Binary primitives is undefined

Beta 2b-4

The TAIS Beta Environment process has access to the Duplex primitive

The Duplex enables the bi-directional streaming of primitives. When the Duplex is first created it is sealed, meaning it can't transmit anything until it is fully connected to its destination TBE process or has been chained to another Duplex that is ultimately connected to a TBE process.



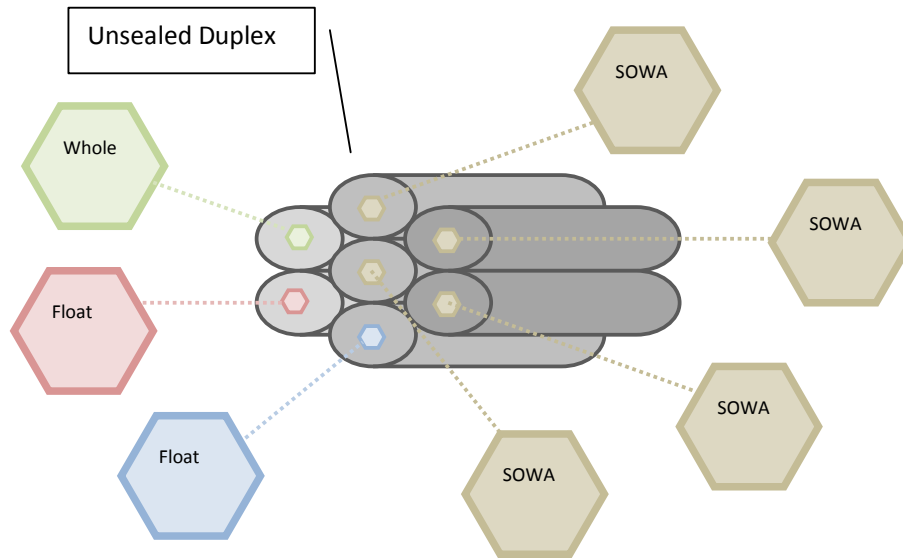
Once connected, the seal can be removed. Various primitives can be transmitted down the Duplex, each in its own channel so that ordering is preserved. For example, if the following writes are made in this order:

1. Float of **25 . 3**
2. Float of **54 . 8**
3. Whole of **9**

and, when the destination reads (in this order)

1. Float it gets **25 . 3**
2. Whole it gets **9**
3. Whole it blocks

The Duplex is a complex primitive because it can contain any number of primitive streams. This is because the Duplex primitive allows for any number of sub Duplex primitive streams to be spun off.



Criteria of abstraction

- 1@B2b-4 A Duplex can only be used once its endpoints are connected to TBE processes or another Duplex that is ultimately connected to a TBE process end-point.
- 2@B2b-4 The data transmitted on the Duplex is only as secure as the network infrastructure that supports it
- 3@B2b-4 The bit encoding of primitive transmission is undefined
- 4@B2b-4 The Duplex must support writing and reading of all other primitives including Float, Whole, Execute and SOWA each on an independent channel
- 5@B2b-4 When a SOWA is written to a Duplex, the Duplex will subdivide its available bandwidth to accommodate the new Duplex channel

The Reference-Membrane

Any data transmitted via the Duplex primitive will automatically be filtered with a Reference-Membrane.

The Reference-Membrane is a powerful tool whose promise is that no references on the destination (receiver) will identify any Set that exists on the source (sender), even though Sets transmitted across the Reference-Membrane retain their integrity relative to each other. The Reference-Membrane allows Static-Memory to pass between TBE processes but not Referential-Memory.

Put another way, the Reference-Membrane will make a copy of the Set as it passes from source to destination. Once at the destination, any cached copy or shared store of SOWA and Execution images may be used for retrieving these primitives on the destination.

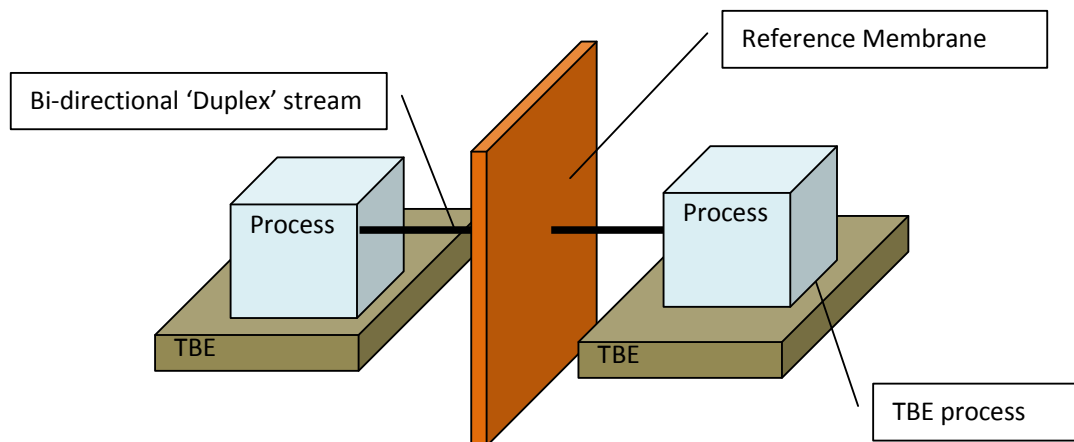


Figure 6b, the Duplex primitive operates as a Reference-Membrane

Beta 3a

All communication between TAIS Beta Environment processes must pass through a Reference-Membrane

Criteria of abstraction

1@B3a All Set references passing through a Duplex must be copied so that no destination reference matches any in the source Set.

The purpose of hiding references is to prevent information or software processes that are communicating from being able to directly affect each other's state. In the figure below, it can be seen how the Reference-Membrane allows data sharing without enabling referencing, yet the Set structures are fully preserved.

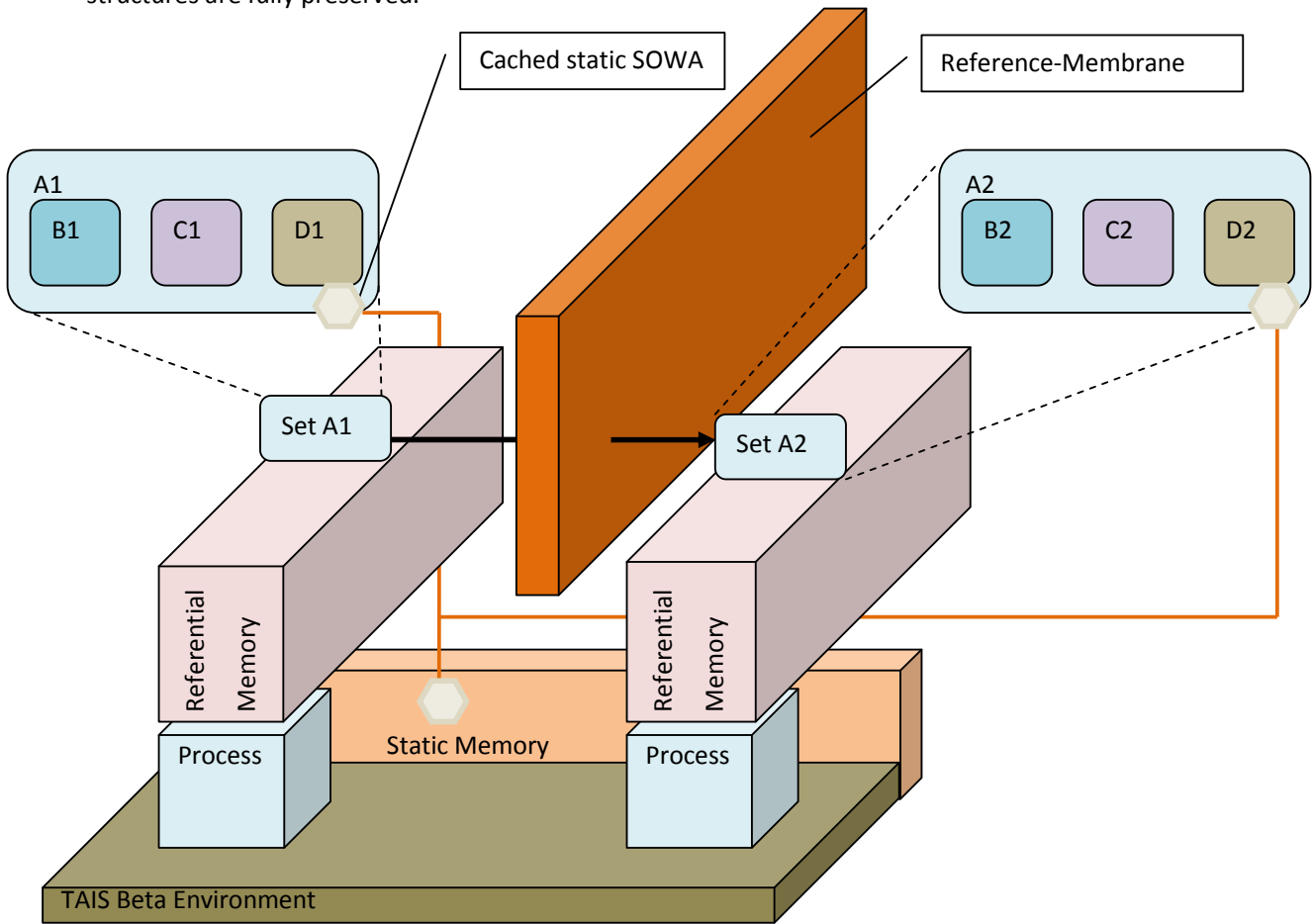


Figure 7, 2 process use a Reference-Membrane to communicate

Beta 3b

All communication between TAIS Beta Environment processes must occur via a Context-Reduction-Engine

A Context-Reduction-Engine (CRE) is a mechanism executing within the TBE process that prevents an indirect remote attack arising across a Reference-Membrane (Duplex primitive) communication channel.

Although the Reference-Membrane prevents one process from accessing another (direct attack), it doesn't stop a backdoor or a bug from being used to disrupt, retrieve information or gain control of the process (passive attack).

For the passive attack to work, the normal behaviour of interpreting communication across the Reference-Membrane must fail when triggered by a specially encoded transmission. In its innocent form, certain encodings would otherwise result in buggy behaviour. In its attack form, this buggy behaviour may be able to cause the process to behave in a way that suits the attacker. Alternatively, backdoors could be deliberately inserted to exacerbate this fault effect to make the attack easier.

The CRE eliminates the passive and backdoor attack altogether by formalising the behaviours that can be committed by each instruction across the Reference-Membrane and reducing the number and capability range of instructions that the TBE process has access to. As a result, the degree of freedom is reduced, but the remaining behaviour is safe.

Of particular interest to high security deployments such as military or government, is that development can be structured into 2 teams. A small trusted team that develops customised CREs and modularises the work to be performed by a much larger and non-trusted team that utilises the CRE to develop units of isolated application behaviour. By forcing the non-trusted (possibly even contractually outsourced) work to operate through the customised CRE, backdoors are eliminated as the code has zero dependencies and can only access the limited CRE functions.

CRE Operation

The CRE works by breaking all communication into a series of micro-instructions called Context-handlers. Every Context-Handler available must perform a single narrow function that can be formally shown to produce safe behaviour without expanding the behavioural scope. An operational information deployment consists of two CREs operating across a Reference-Membrane as:

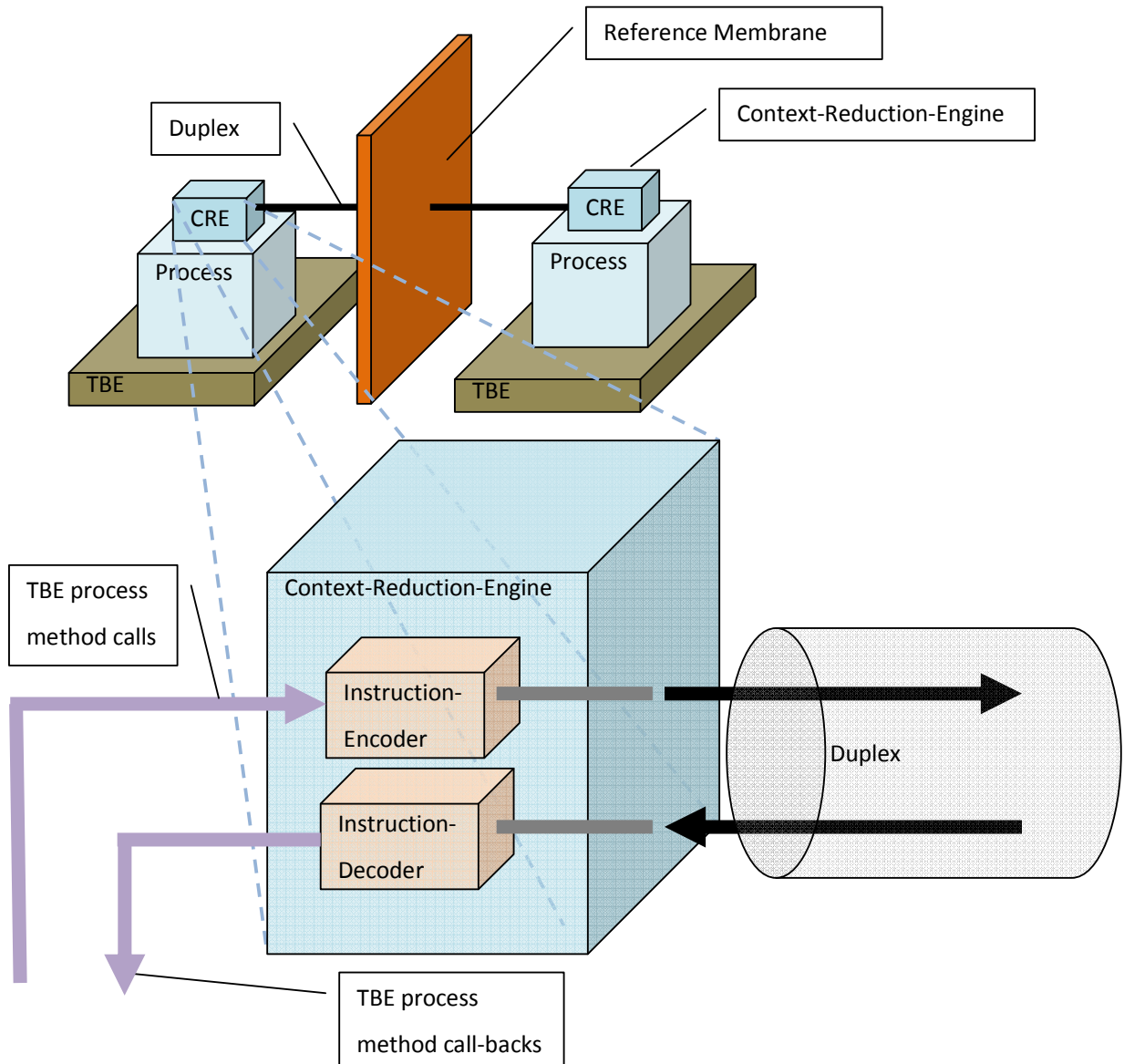


Figure 8, processes can be protected if all communication is conducted via a suitably limited Context-Reduction-Engine.

The CRE is only effective if all externalisation and communication is driven through it. The engine converts method calls, call-backs and their complex parametric objects into a series of isolated instructions. Each instruction is narrow in scope¹³ and can be audited independently by a small trusted team.

This means non-trusted developers (contractors) are free to implement new software that uses the CRE for all TBE process externalisation. By isolating non-trusted developer code within its own TBE process that communicates with other project components via a CRE, it is not possible to install a backdoor or accidentally encode software vulnerabilities. Non-trusted code should not have access to any more CREs than is necessary for it to process its modular function.

Criteria of abstraction

1@B3b	The TBE process must engage all communication with other processes via a series of Context-Handlers
2@B3b	Each Context-Handler must be scope limited so that all possible behavioural outcomes can be shown to be safe
3@B3b	The Context-Handlers may not contain any errors that could degrade security
4@B3b	Non-trusted code must reside on its own TBE process
5@B3b	Non-trusted code must utilise a minimum number of Context-Handlers

Context-Handlers should be limited in their behaviour. The following behaviours can be automatically checked at design time via a formal software verification process:

- Check that every communication established by TBE process is managed through a CRE
- Check that Context-Handlers themselves do not start a new 'virtual' thread if threading is emulated by the TBE single threaded environment

¹³ A bug in a narrow context definition cannot be exploited beyond the outer boundary of the predefined Context-Handler. For example, compiling a graphic routine to draw a straight line between two points cannot be used to retrieve other data onscreen. Any bug would produce dysfunction behaviour in this narrow context.

- Software that invokes a call to insert a Set reference into any other Set whose source has been received by the Instruction-Decoder should be flagged for intense manual analysis and formal verification proof
- Context-Handlers should ‘reboot’ after processing each instruction
- Modular TBE processes using CREs should ‘reboot’ after each transaction

5@B3b is an example of a Factor that would be extremely hard to fully achieve in practice. The TSR induced by 5@B3b will depend on:

1. how modular the non-trusted code is
2. the number of CREs required to operate
3. the breadth of functionality of each CRE

In reality, whatever risk accumulate in the factor can be reduced by not outsourcing the TBE process and by using formal accreditation methodology to reduce the risk further – but the clear advantage with TAIS here is the sheer cost and speed of implementation if accreditation is avoided.

Since the TBE inherits the TBA, the accumulated Criteria of Abstraction for the TBE is as follows:

The TBE Criteria of Abstraction

Run-Time Factors	Design-Time Factors
Alpha 1a, Alpha 1b & Alpha 1c, Alpha 3	Alpha 2a, Alpha 2b, Alpha 2c, Alpha 2d, Alpha 2e
Beta 3b	Beta 1, Beta 2, Beta 3a

Step 3

Secure Grid communication

GAMMA ENVIRONMENT

Although the TAIS Beta Environment (TBE) enables secure communication between processes, it is severely constrained because the communication must be established at TBE boot-time when the Duplex primitive is passed in as a parameter.

The TAIS Gamma Environment (TGE) dramatically expands the capability of the TBE by establishing a Virtual-Operating-System (VOS) that can handle complex functions including the management of loading new processes and facilitating routing in an open and unbounded distributed peer-to-peer grid infrastructure.

The security promise of the TGE is that integrity will be maintained as it participates in the open grid even if its direct partners are compromised.

These additions result in the following emergent properties:

- Each TGE is a unique Node with a controlling interface managed by a Node owner
- The owner of each Node selects the peer Node partners and can profit from the resources offered to public processes that utilise it.

The resulting network is physically open but secured because any process¹⁴ can move to any Node. The process must both reserve and pay for resources consumed and cannot attack the host.

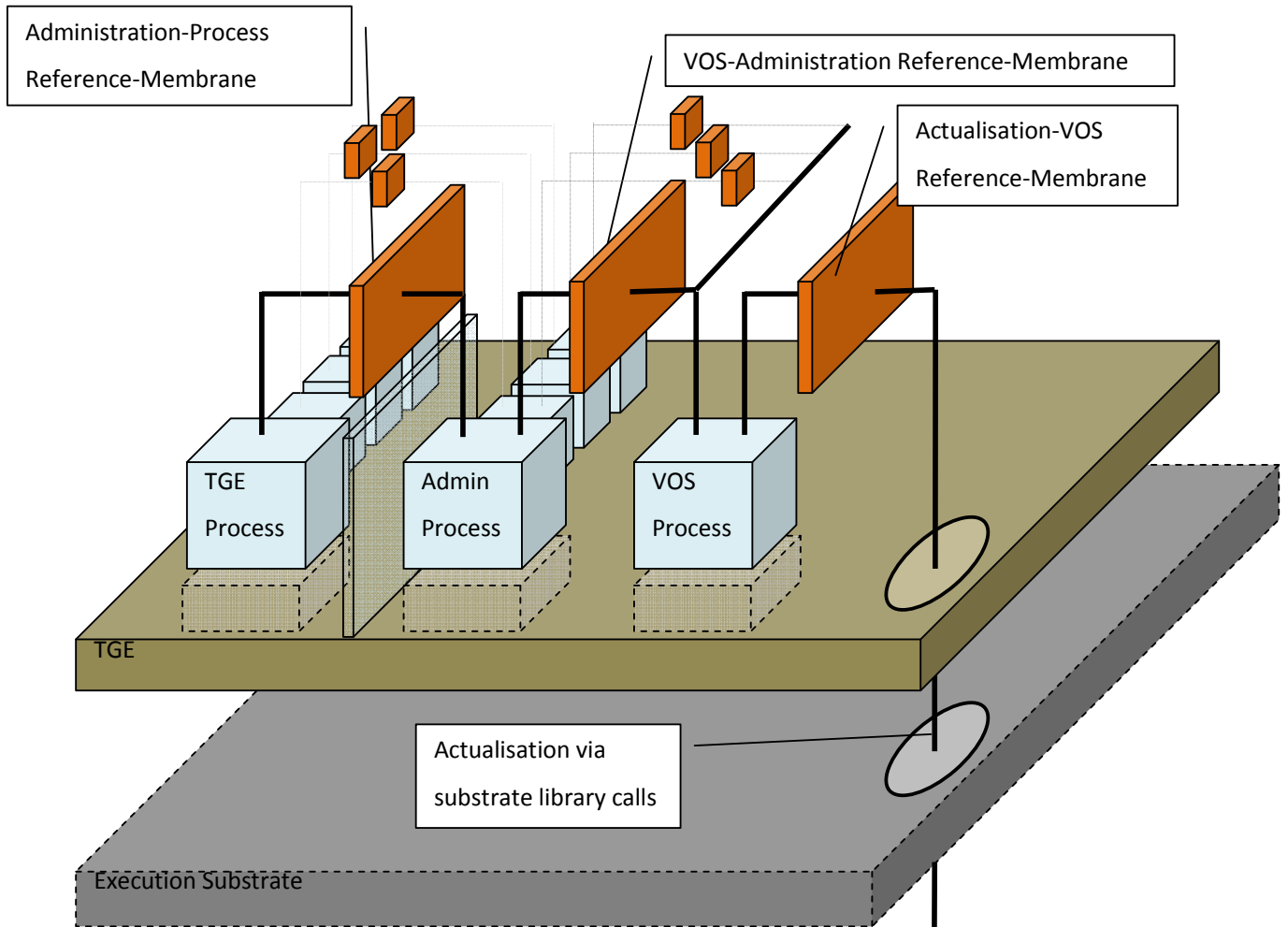


Figure 9, the process and stack layered view shows how the VOS layout prevents attacks from both the process and backdoor vulnerabilities that may have been installed by spies masquerading as developers for the nation state

¹⁴ Transmission of the Currency primitive enables money transfers in the Gamma Environment

Operating System complexity

The establishing of a Virtual-Operating-System (VOS) in the TGE introduces a number of significant security factors:

Non-trusted developers

Due to its inherent complexity, the VOS codebase must be designed to be maintained by many developers, some of which may otherwise attempt to install a backdoor onto the Node. Alternatively, developers will from time to time introduce errors in their code. Regardless, whatever code exists in the VOS, it must not compromise the Node.

The TGE is designed to prevent unauthorised accessed to information during VOS malfunction and its graceful degradation ensures that the TGE Admin-Process disruption will not compromise the information it manages.

The TGE makes the *fail safe* security guarantee with non-trusted and unaccredited software by encapsulating the VOS at 3 distinct points and separating each area of scoped logic and state data with two pairs of CREs.

Let's unpack that previous paragraph in more detail and see how this counter-intuitive mechanism produces secure code shrink-wrapped in a behavioural straightjacket, regardless if it was written by an enemy of the state.

Firstly, take a careful look at the diagram on the next page, then follow the explanation that follows to understand how TGE processes are protected against a faulty VOS while the VOS is protected from attack by the TGE process itself.

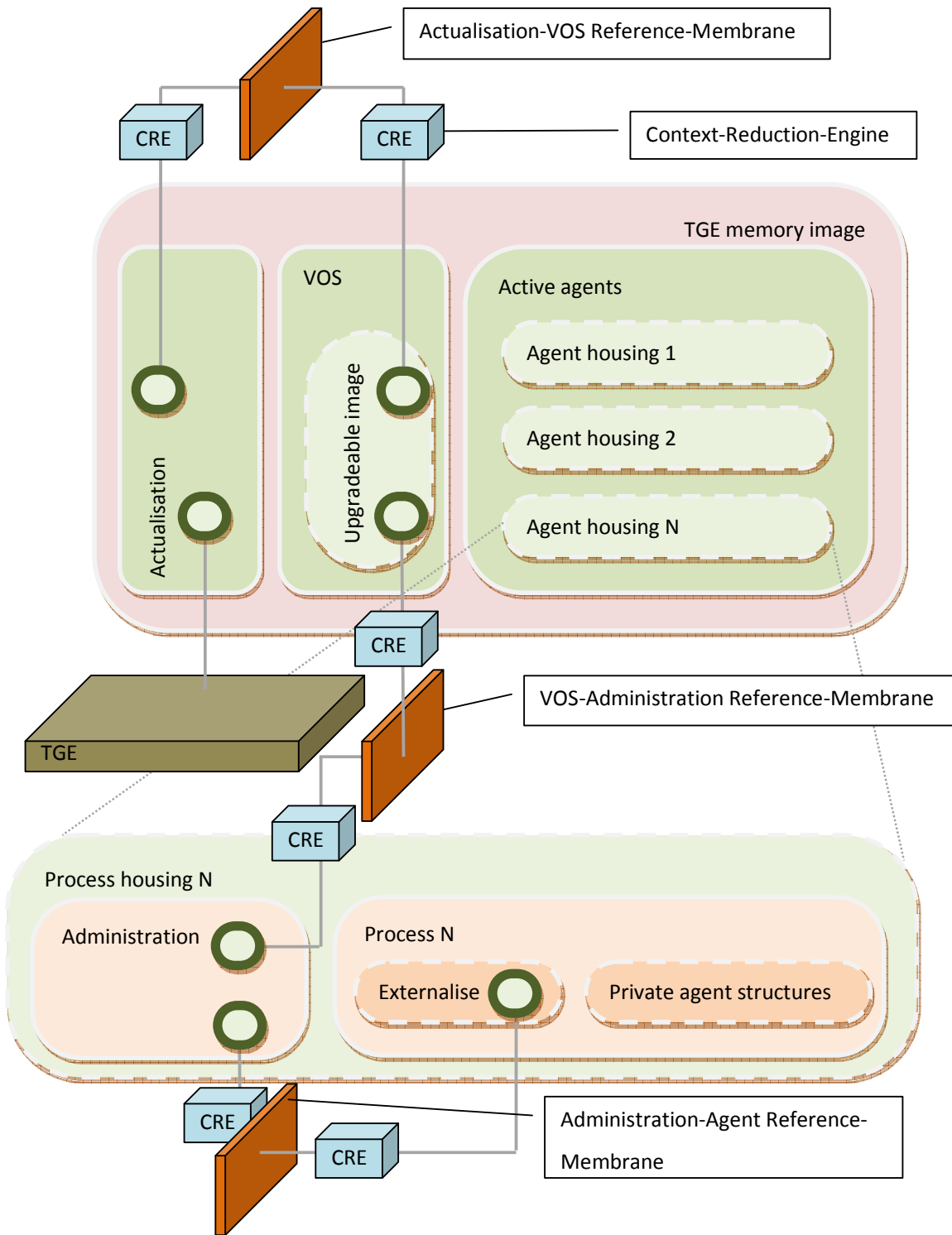


Figure 10, A Referential-Memory view of how three pairs of Context-Reduction-Engines ensure 3rd party OS code can operate within safe parameters.

The memory layout of the VOS is indeed complex, but on inspection it can be verified that integrity is preserved for:

- TGE processes
- Administration
- VOS
- The TGE itself
- Substrate

Each of the above areas is separated by a pair of CREs and a Reference-Membrane.

Inside each CRE, only the Instruction-Decoder can pose a security risk, as it is the only component within the TGE Process capable of causing a change of state in memory. As a result, the level of scrutiny and criteria of abstraction will vary between TGE processes, Administration and the VOS.

Administration-Agent Reference-Membrane

Each TGE process is booted with two additional processes operating within isolated Referential-Memory. The Admin process is dedicated to serving the TGE process, and as such, is non-critical. The Admin process is designed to manage the vast majority of TGE process support services and is therefore designed to be developed by a large non-trusted team that may utilise unaccredited 3rd party components.

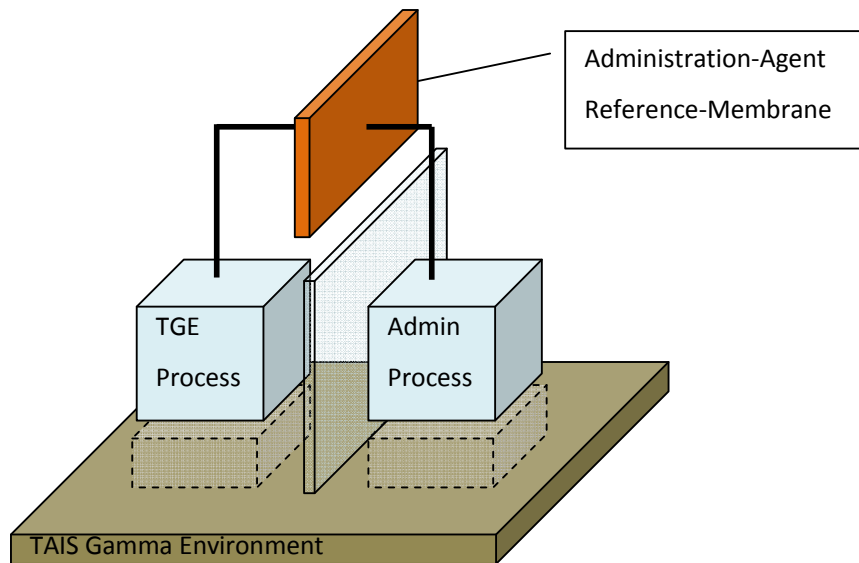


Figure 11, the TGE process can only externalise via its dedicated Admin-Process

VOS-Administration Reference-Membrane

The Admin-Process that may cause failure only for the single TGE process that somehow activates a dormant vulnerability or backdoor. In contrast, the Virtual Operating System (VOS) is shared by all the TGE processes on the Node and as such has rigorous security criteria.

The VOS must be written by trusted personnel and requires accreditation. Complex as it is, the TGE is forgiving with regard to the architecture of the VOS. Accreditation is an expensive and cumbersome process, so the VOS is designed to be built once in its life-time and is also relatively small in comparison to the majority of the logic occurring within the Admin-Process. The VOS is primed for the Common-Criteria accreditation process.

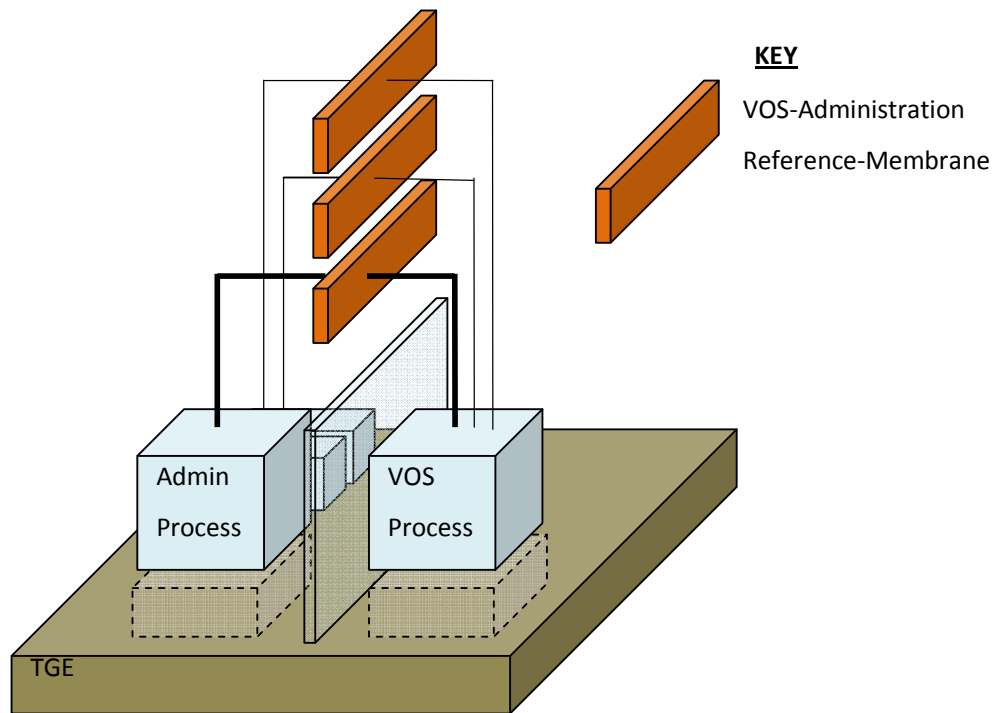


Figure 12, every Admin-Process shares access to a single VOS-Process

The instructions moving across the VOS-Administration Reference-Membrane should be low-level and act as an API to allow complex and dynamic Admin-Process logic to enact service requests for the TGE process.

Actualisation-VOS Reference-Membrane

The Actualisation-VOS Reference-Membrane is designed to protect the TGE and substrate from a theoretical fault occurring in the VOS. Even though the VOS should be accredited, this last Reference-Membrane guarantees that multiple TGEs run from the same substrate will not affect each other, particularly in the scenario where a TGE is running with non-trusted VOS code.

The kind of instructions being managed by the TGE as it processes the Actualisation-VOS Reference-Membrane instruction stream includes:

- Propagation¹⁵ of a TGE process to another adjacent TGE Node.
- TGE processes can transmit primitives¹⁶ to other processes residing on an adjacent partner in the same way they would communicate with a local TGE process on the same Node

¹⁵ Processes executing on the TAIS implementation can create executable processes on any partner Node provided the source process contains the Execution primitive in its scope.

¹⁶ Node owners need to select their partners because primitives will need to be recompiled depending on the implementation of the target TGE. See Heterogeneous codebase in the next section.

Heterogeneous codebase

The TGE is designed to operate as a distributed open unbounded peer-to-peer grid network. This has the profound advantage of allowing for unforeseen requirements without locking the architecture down in the way HTML and TCP/IP has dominated the network paradigm since its adoption (2011).

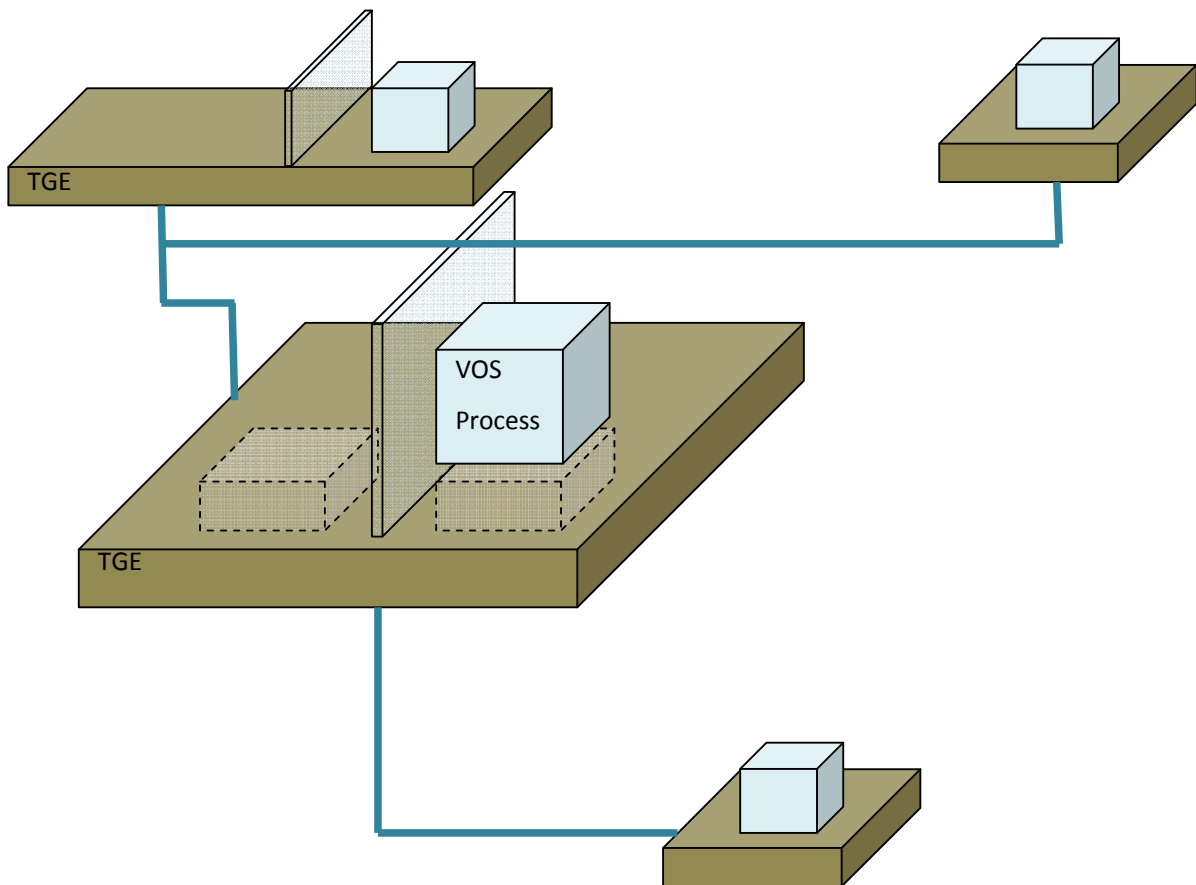


Figure 13, TGEs can connect together to form a peer-to-peer network

An unbounded grid means:

- The grid has no owner or centralised management (unlike the DNS system)
- Anyone can create a TGE Node and join
- Anyone can create a TGE process that can live anywhere on the grid
- The Node is free to be implemented in any way¹⁷.

It is the last bullet point above (allowing architectural freedom) that introduces a technical requirement similar to cross-compilation and is listed under Criteria for Abstraction¹⁸. The TGE Nodes that choose to link to each other must be able to support the transmission of primitives between each TGE Node during Duplex communication.

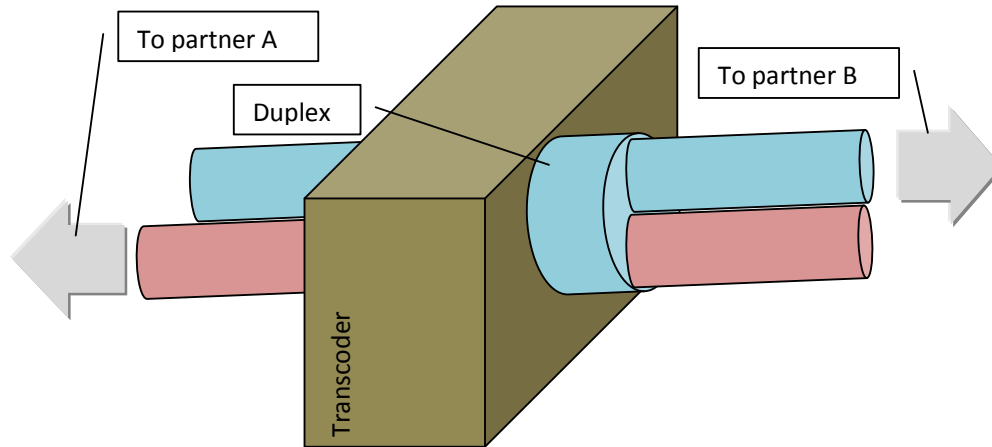


Figure 14, a Transcoder translates primitives between Nodes for each Duplex connection

¹⁷ It is not even a requirement that the Criteria of Abstraction are met because as will be seen in the TAIS Delta Environment (TDE) next, TGE processes can choose the Node they reside on, so rating services or absolute knowledge systems (such as private Node installations) will avoid non-trusted Nodes that do not comply with TAIS and avoid using them as routes.

¹⁸ The TGE Node must identify and be able to connect to specific versions for which it is compatible.

The primitive conversions that occur during Duplex transmission between Nodes of different architectures is purely mechanical and occurs as follows:

- Whole and Float numbers are simply remapped to their bitwise symbolic equivalents
- Each SOWA block is remapped the same as above provided it is a Whole or Float while Binary flagged blocks remain unchanged.
- Execution represents a cross-compiled code sequence for the target environment. Since there are exactly zero dependencies (no library calls), this is straightforward.

The freedom to implement different architectures and Operating Systems on the same computing infrastructure while retaining data interoperability is powerful because it enables forward and backwards compatibility at a computational substrate level (as opposed to semantic level that occurs between applications as they communicate – say through XML).

Regardless of the advantages, it does lead to geopolitical issues that are beyond the scope of the theory directly, but can be managed within specific implementations of TAIS. Such issues arise naturally because the VOS remains undefined. As a result, each nation state can produce variants of the VOS that will lead to interoperation issues at an information level.

One solution that addresses these geopolitical issues is the Interoperable Information Infrastructure (3i) and an experimental (at 2011) specification (BI3 from www.blueicon.com) and implementation (Bluebrick©) are available as free and open resources for academic investigation.

Ecosystem functions

Another consequence of design for an unbounded grid means that TGE processes can migrate virally between Nodes. However, these mobile software units are rendered impotent because the basic economic principles for any successful software eco-system must exist within the VOS in order for the deployment to be economically viable.

The TGE process is accepted onboard as a harmless customer and contributes to the grid ecosystem with:

- Node routing services
- Node rating agencies
- Node currency brokering¹⁹
- Node resource scheduling

The specifics of these types of services are mechanical and captured easily as a class of behaviours operating on the TGE and so are beyond the scope of the TAIS objective of information security.

¹⁹ It is expected that the VOS will issue virtual currency tokens in exchange for resources purchased on the Node by TGE process customers. Various reservation models may be used, but a prepaid and limited accessed on demand is a minimal recommendation

Functional expansion capability

Although the TGE allows for almost any kind of implemented architecture and configurable VOS, it still conforms to an abstract framework. There are some situations whereby local substrate services may be required, and ordinarily, these would be off limits to a TGE implementation.

The TGE allows for unrestrained access to the Execution Substrate and as such introduces an unknown security risk.

In the event that substrate services are utilised, then TGE Node security must be downgraded because non-virtualised code executing at the substrate level is servicing a TGE process directly via the implementation's expansion framework. This capability is most likely to be used on isolated Nodes that need specialised functionality, such as accessing legacy systems or executing control instructions for dedicated hardware interfaces.

However, as shall be seen in the TAIS Delta Environment (TDE) next, these Nodes that pose an undefined risk profile should simply be avoided for storing information.

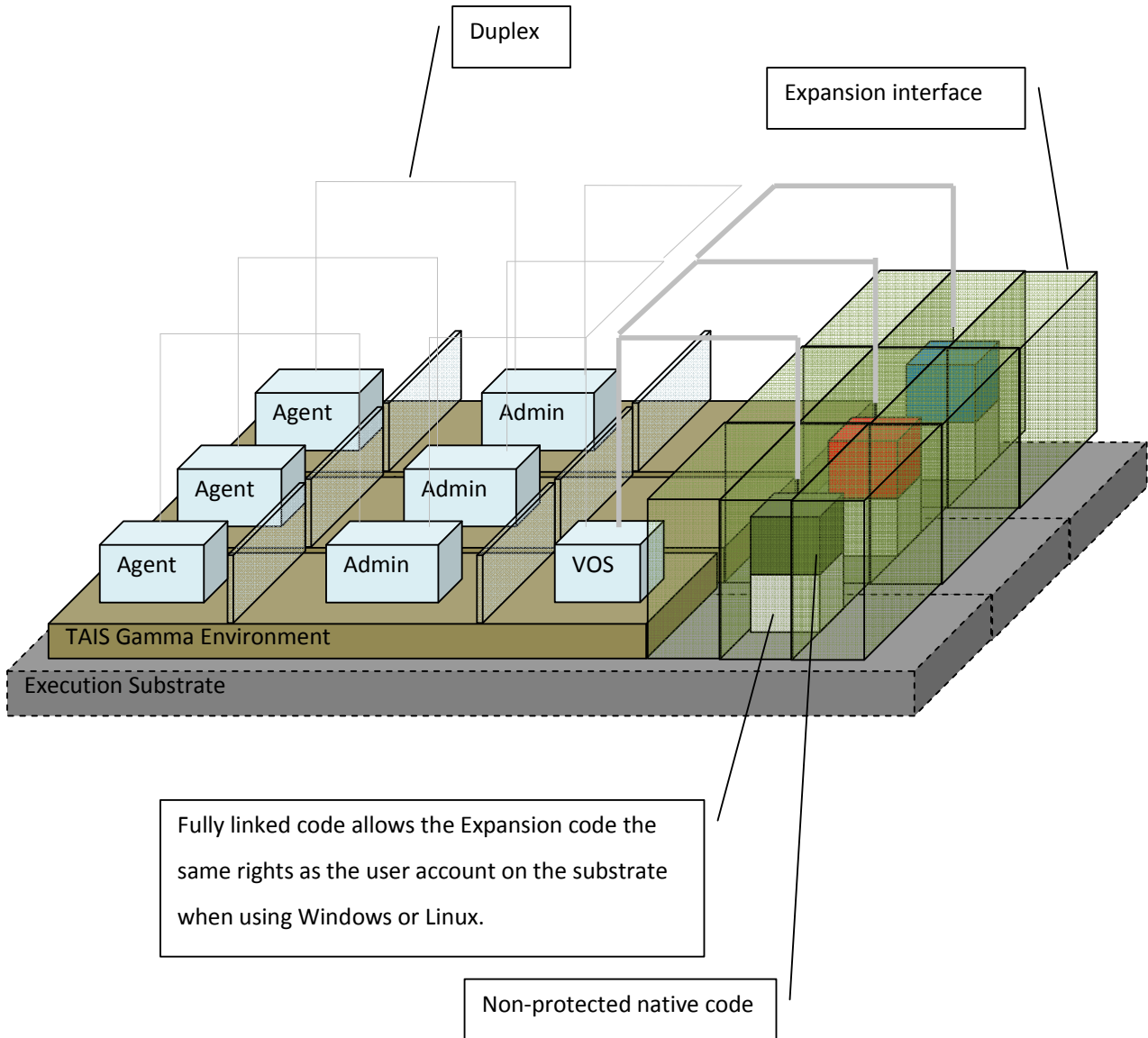


Figure 15, the expansion interface introduces an unquantifiable security risk

Primitives and Factors

The TGE adds only one new primitive type, Currency:

Primitives in the Gamma Environment

Primitive	Description
Execution	Determines the complete execution sequence and is not accessible introspectively
Float	N-Bit Floating point number
Whole	N-Bit integer number
SOWA	Static Once Write Array (SOWA) is an N-Bit array that cannot be modified once established
Set	A referential Set structure that can bind each class of primitive only once
Currency	A local currency unit relative to the Node

The TGE Factors

1. The Virtual-Operating-System (VOS) must not compromise the TGE process
2. The Reference-Membrane and CRE pair must be used for VOS and TGE process communication
3. Each Node partner must be trusted (or have a pragmatic borrowing limit) to clear monetary transfers externally on behalf of all TGE process transmission of the Currency primitive.

TAIS Security Factors for the TGE

Run-Time Factors	Design-Time Factors
Alpha 1a, Alpha 1b & Alpha 1c, Alpha 3	Alpha 2a, Alpha 2b, Alpha 2c, Alpha 2d, Alpha 2e
Beta 3b	Beta 1, Beta 2, Beta 3a
	Gamma 1 & Gamma 2
Gamma 3	

Step 4

Secure process communication

DELTA ENVIRONMENT

The TAIS Gamma Environment (TGE) enables a physically open grid network where the infrastructure remains secure; however, TGE processes operating across this environment are still prone to the Malicious Host Problem, meaning that if they reside on an insecure Node, they can expect any data they possess to be fully readable and modifiable. This data could be in the form of information content, prepaid monetary tokens, symmetric encryption keys or any other encoding.

The security promise of the TDE is that information and software can operate with predefined security on an open grid network, provided the Nodes selected for computation are fully secured themselves.

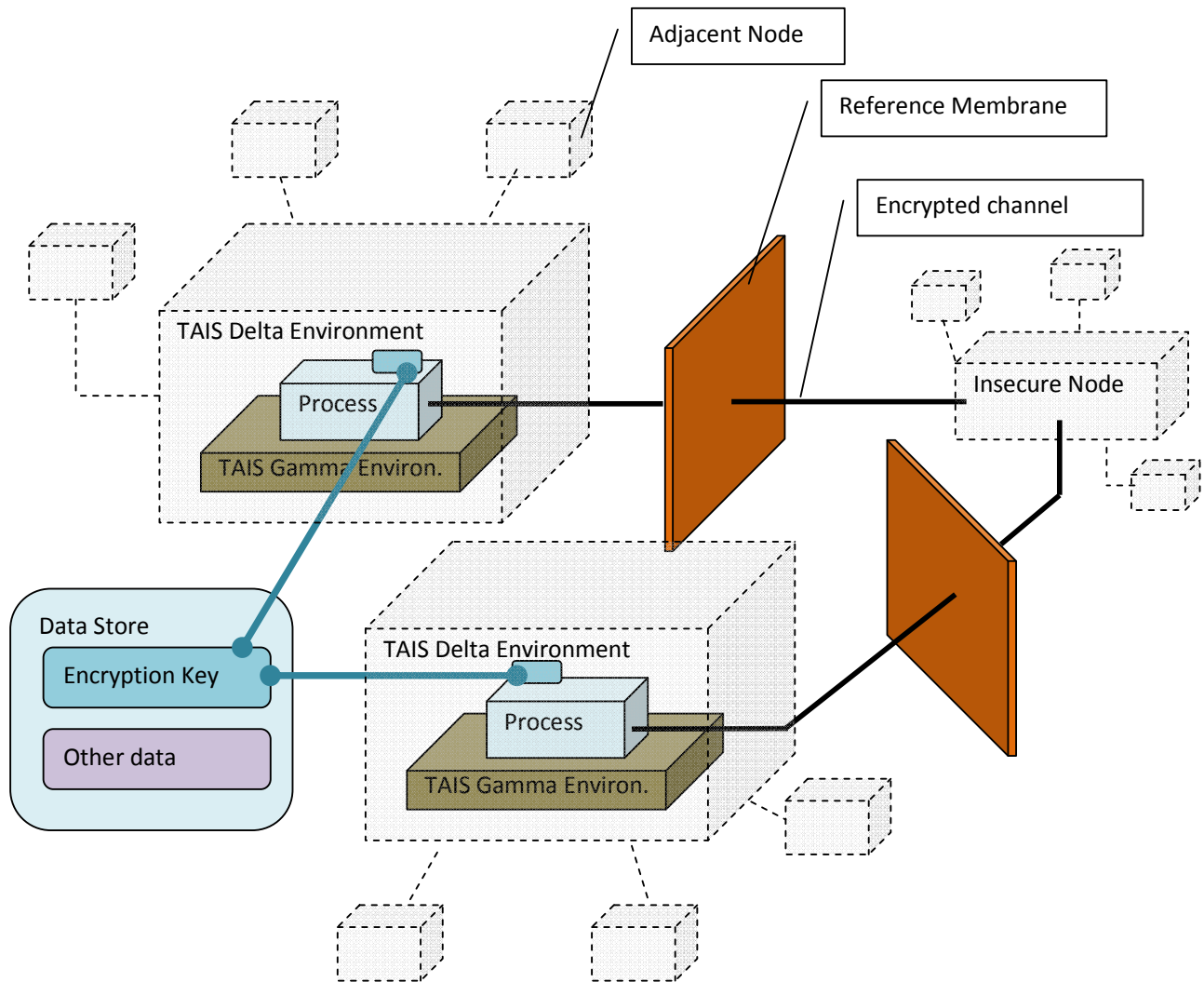


Figure 16, the TDE requires each Agent to be fully responsible for the host on which it executes; however, by using encryption, less secure Nodes can be used providing the encryption keys are never exposed as clear text on a non-trusted Node.

There are 2 principles governing the security of the TDE:

- Node selection
- Encryption

Node selection

The TAIS Delta Environment (TDE) ensures that TDE processes proactively select the Nodes on which they reside. By operating from only trusted TGE Nodes, TDE Processes can be assured that they are never compromised.

Selecting a secure Node includes preventing physical access to it to avoid disrupting the TGE factors already established.

If 3rd party software is used to select routes, this too should be trusted.

Another key factor for Node selection is the deliberate avoidance of backups on the Node.

Redundancy is managed by the TAIS Epsilon Environment (TEE), so it is important to ensure no additional route (backups) to obtaining the information exists. Trusted Nodes should therefore never schedule a system backup. In this way, TDE process data remains current. When items are deleted in the TEE, they are fully destroyed and sensitive information that is encrypted is not prone to future brute force attacks (when more processing power becomes available) in the event that the backup itself has become compromised.

Encrypted transmission

Non-trusted Nodes can be used as transit gateways provided that all data is symmetrically encrypted²⁰ and the keys are never processed on a non-trusted Node or are transmitted in an unsecured way²¹.

TDE network classes

Networks have two aspects to their form – firstly they have a physical aspect where the TDE process that instantiates them must select Nodes on which to operate (which is the concern of TDE).

Secondly, they have a logical aspect in how they are structured once they are established on the Node (which is the concern of the TAIS Epsilon Environment in Step 5 next).

Two network scenarios are likely to dominate:

Private networks

It is impossible to prevent people establishing private networks of any architecture and the TDE reinforces this principle by allowing private unreachable networks to operate on grid. Corporate, personal data stores, militaries and governments are most likely to establish private networks that share parts of the infrastructure with public networks (in stark contrast to the networking paradigm at 2011).

²⁰ Asymmetric (public/private key) encryption algorithms should be avoided because they rely on the non-existence of an algorithm to quickly factorise large prime numbers, even though such an algorithm is clearly not intractable. Improvements in factorisation may arise accidentally or without design, such as in the case of the remarkable Bailey-Borwein-Plouffe formula that is used to calculate Pi that inexplicably appears to resolve the answer. A recent example of how quickly an organised modern team can crack difficult math problems is the case of Ken Ono and his team who were funded by the American Institute of Mathematics (AIM) in 2011 and solved the 200 year old problem of Partition Numbers that evaded Euler, Ramanujan and Rademacher who all worked on the problem. Although useful, asymmetric encryption systems provide the illusion of a secure TDE Security Factor 4 (see factors). In an overnight catastrophe, the entire security premise could evaporate after a mathematical breakthrough. Worse, if this discovery occurs at the nation state level and is kept secret, administrators will not even be aware that their systems are insecure. In sharp contrast, symmetric encryption is likely to become 'weak' gradually over time as brute force attacks improve, but since all Nodes are dynamic and incur no backups, keys can be strengthened over time to match brute force attacks.

²¹ The problem of moving unencrypted keys for symmetric encryption algorithms through the network is beyond the scope of TAIS as there are many mechanisms including a one-time pad. However, secure and trusted routes over existing encrypted channels are likely to be shown to be equally secure for key delivery in the deployed network.

Although it is entirely possible for the Nodes that support these private networks to also host public shared data securely, some ultra high security implementations may chose to prevent sharing of Node resources with public data in order to fully eliminate any potential residue risk emanating from the TGE runtime factors.

Economic networks

Information that is to be shared by people on the grid has a number of properties that differentiate it from private networks.

Routing-Services

As information is stored in a distributed fashion, public routing services (with VOS control interfaces) are likely to emerge. These services would located TDE process on Nodes as well as select various peer-to-peer routes to connect them – each independent fixed route can not only guarantee bandwidth delivery, it can increase traffic and redundancy simultaneously.

Information interoperability

Unlike private networks, information in a public network must be rendered in a universal way that never changes so that it always available for use. The TAIS Zeta Environment (see TZE) specifically caters for this requirement by introducing a Personal-Render-and-Capture primitive that is updated between TGE versions in the same way as all the other primitives are (see TGE) as they are re-encoded between Nodes. This allows information formats to be changed while retaining forwards and backwards compatibility.

Charge for use

In order to accept unknown processes on board, all resources need to have a charge applied so as to prevent the overloading (freeloading) and abuse of free resources that would otherwise exist on the Node.

Currency system

Although financial payments for Node resources can be linked into the offline economy, it is most likely that the VOS will include the capability to issue digital monetary tokens that can be used to pay

for disk, memory, processor and bandwidth consumed on the Node, which may ultimately constitute an independent monetary system.

Reservation

In a similar way that resources usage is protected with charging, a reservation system balances the demand times for processor in order to prevent overloading as the natural price point changes. Many systems are possible, but a single reservation system would need to be agreed to for an economic network to operate effectively. It is likely that standard resource reservation, possibly with the requirement of being paid upfront coupled with a capped on demand emergency resource cover scheme would likely dominate.

Enhanced economic model

It is likely that the designers of economic networks will take this unique and pressing opportunity (at 2011 economic gloom has been lingering since 2007) to optimise the economic engine into a more efficient configuration. It is likely that sharing of public information without the permission of the creator will be allowed while guarantying a royalty payment for its one time use. This public software is likely to have a free and paid version to enable it to be rendered on all systems.

Information delisting

Geopolitical requirements for information control will likely see information being banned where it violates copyright, patents, involves illegal information such as child porn, drug trading or political propaganda such as a WikiLeaks scenario or anti-state propaganda required by some countries.

Taxation

Revenue generated on an economic platform will be taxable without loss to evasion. Global co-operation can be strengthened (such as CO2 targets) by deducting taxable revenue where other non-network agreements are in shortfall.

The TDE Factors

1. The TDE process must select a trusted Node on which to execute
 2. The TDE process must encrypted data that is to pass through non-trusted Nodes
 3. The TDE process must utilise all encryption keys only on trusted Nodes
 4. The encryption algorithm must be secure
-

TAIS Security Factors for the TGE

Run-Time Factors	Design-Time Factors
Alpha 1a, Alpha 1b & Alpha 1c, Alpha 3	Alpha 2a, Alpha 2b, Alpha 2c, Alpha 2d, Alpha 2e
Beta 3b	Beta 1, Beta 2, Beta 3a Gamma 1 & Gamma 2
Gamma 3	
Delta 1-4	

Step 5

Control information accessibility

EPSILON ENVIRONMENT

Although the TAIS Delta Environment (TDE) allows information to be stored securely on the open grid network, it is not truly useful until it can be accessed by people. The TDE is ideal for system to system communication, but lacks the ability to put information in scope and allow personnel to access different information governed by that scope.

The TAIS Epsilon Environment (TEE) incorporates a novel information database founded on Set Theory that prevents accidental or deliberate dissemination of information.

TEE allows for the following kinds of operation scenarios:

- A politician could send a message to a member of their caucus by using TEE²², but specify a larger group beyond which the information could not be disseminated.
- The Secret Service could delete a document with the knowledge that there was no digital copy remaining; yet re-instate that information at a later date if required.
- Highly authorised persons can give anyone any document without fear that they may breach information security protocol by accident or negligence
- Spies are unable to forward sensitive state information
- Documents cannot be downloaded or copied as in the WikiLeaks scenario

²² Since TEE does not address the rendering of this information, TEE is unlikely to be used in practice. Any additional external rendering of TEE would adversely affect the security function. However, some dedicated security applications (perhaps legacy systems) could use TEE as a secure transport layer. It is expected that the TAIS Zeta Environment (TZE) will handle rendering of most information.

- Information can be stored in its most logical form without being structured in blocks that relate to broad categories of access
- Information can be referenced in any number of places, so documents can be dynamically updated once and be accessible where-ever they are linked
- Access to a single unit of information can be fully controlled, no matter where it is linked
- Information can expire, self-terminate or be retrieved by the sender if required
- Users can access new application software without having to trust the software developer
- For low-security deployments with a high TAIS risk profile, account intrusion can be managed as sensitive information, money and control panel are not accessible.

The TEE information store requires two mathematical additions to the TAIS Delta Environment (TDE).

[1] The Entangled-Set

The TEE extends the TDE by allowing the Set primitive (introduced in the TAIS Beta Environment) to be extended. Set primitives in the TEE can be *entangled* at run-time by the Agent. The Set will remain entangled for the remainder of its life and from that moment onwards it can be used to store sensitive data safely.

The Duplex primitive naturally acts as a Reference-Membrane for all Sets that are transferred across it. The Entangled-Set takes this further and acts as a referential mirror when transferred through a Duplex primitive. When the Reference-Membrane encounters an Entangled-Set, instead of producing a copy of the reference at the target like it would do for an ordinary Set, it will produce a single Set reference for the entire Entangled-Set. Once it arrives, the destination is free to add to this blank Set. Regardless of the contents of this Set, it will be transformed back to the original Set stored at the source when passed back through any other Reference-Membrane in reverse.

Let's pause there for a moment and examine that more closely.

The entangled Set is a simple but powerful concept that is best explained diagrammatically:

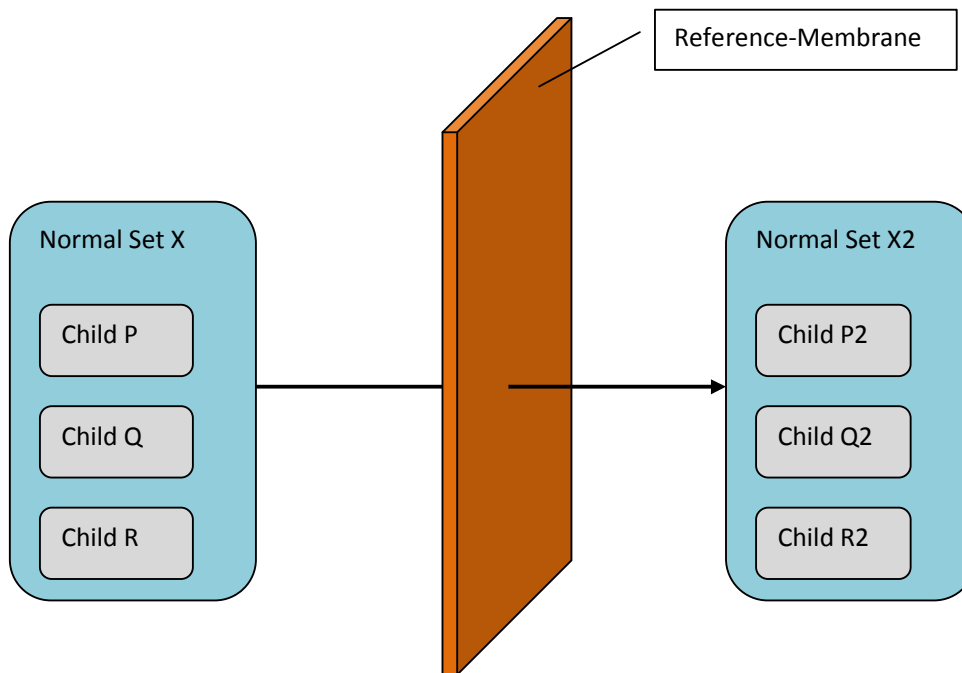


Figure 17, *Normal Set X* is transferred across a Duplex primitive and arrives at its TEE Node destination as *Normal Set X2*, which is a perfect duplicate copy.

In the normal transmission of the Set primitive across the Reference-Membrane, unique references are created at the destination; however, all Static-Memory references are preserved.

When the normal Set is transferred back in reverse across the Reference-Membrane, the process is repeated and a 3rd unique Set of references is created.

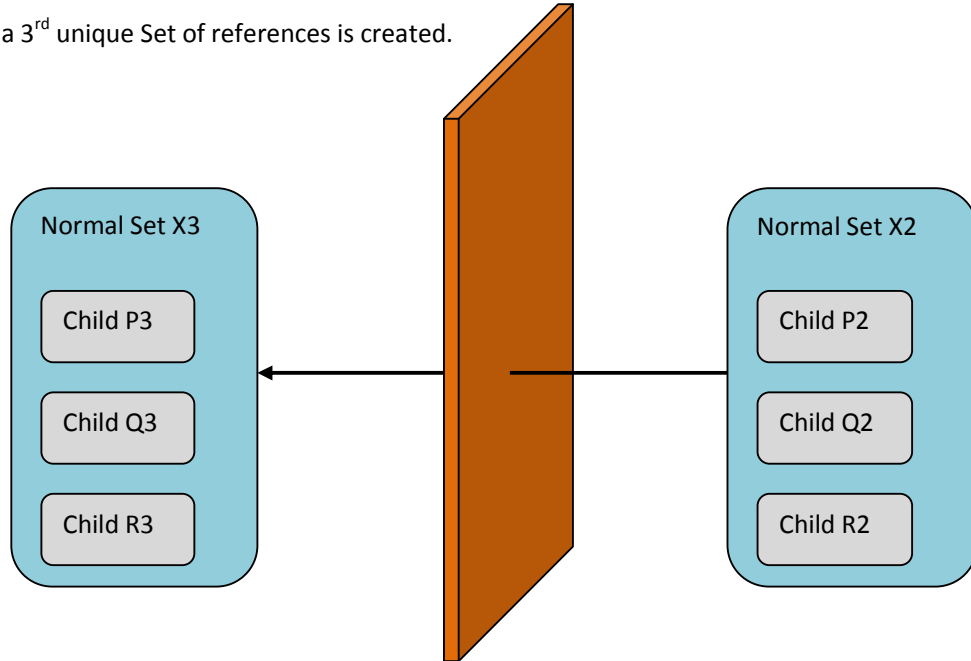


Figure 18, normal Set transmission across a Duplex

However, in the transmission of an Entangled-Set, a single blank Set is created on the first transmission.

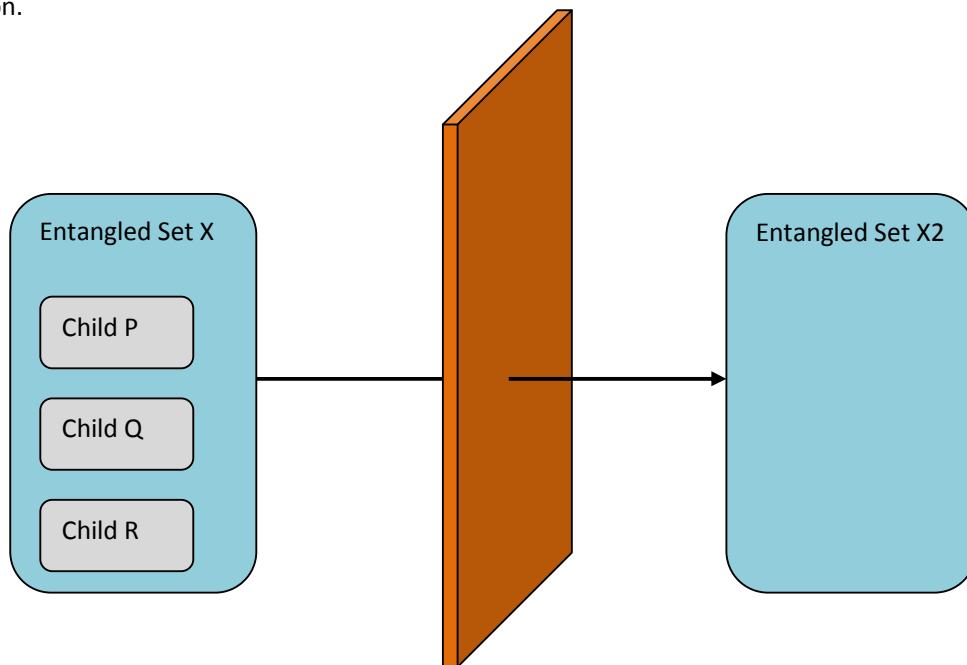


Figure 19, Entangled-Set transmission across a Duplex

This new Entangled-Set is free to accumulate any private data without fear that it will be exposed across the Reference-Membrane. When the Entangled Set is transferred in reverse, the original is presented – so no actual transfer takes place.

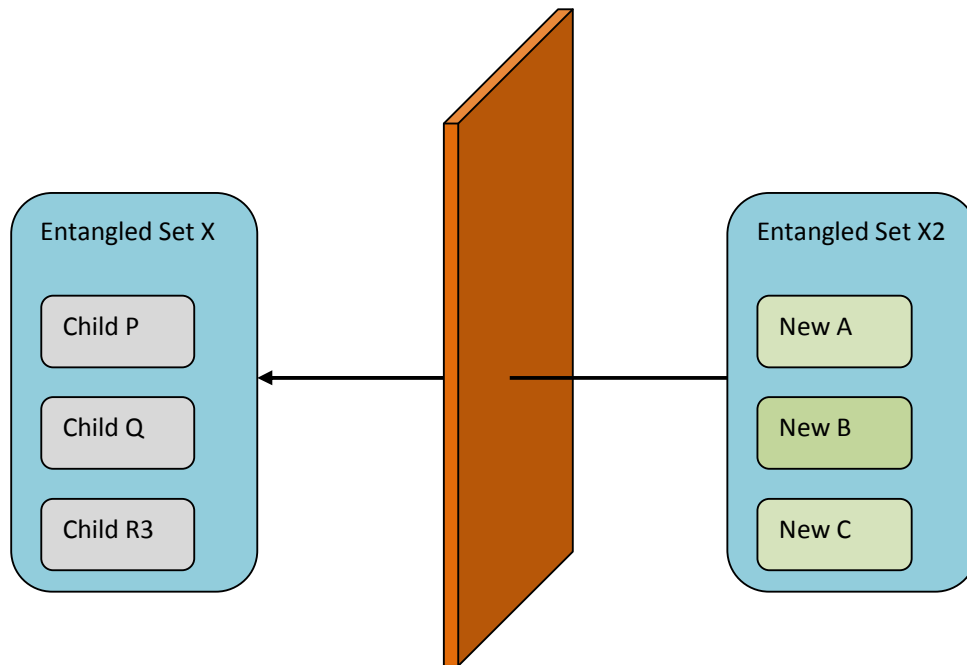


Figure 20, private Entangled-Set storage on the destination

Criteria of abstraction

The Entangled-Set implementation must:

- ensure the Set be able to be transferred across any route to any Node and never collide with an internal representation of any other Set on the grid²³.
- not be reproducible by any peer Node that transmits Set primitives to the local Node. Each Node must be able to verify the integrity of any Entangled-Set prior to use but without being made aware of the hidden secret²⁴ that makes the Entangled-Set unique.

²³ This is important, since normal Sets are simply copies stored locally, whereas Entangled-Sets represent relationships with Sets on external TGE Nodes and therefore are exposed to address space collisions.

[2] Bookmarked-Sets

The Bookmarked-Set is a mechanism to reference Sets. Any Set reference is made with explicit reference to both its Parent and Child simultaneously.

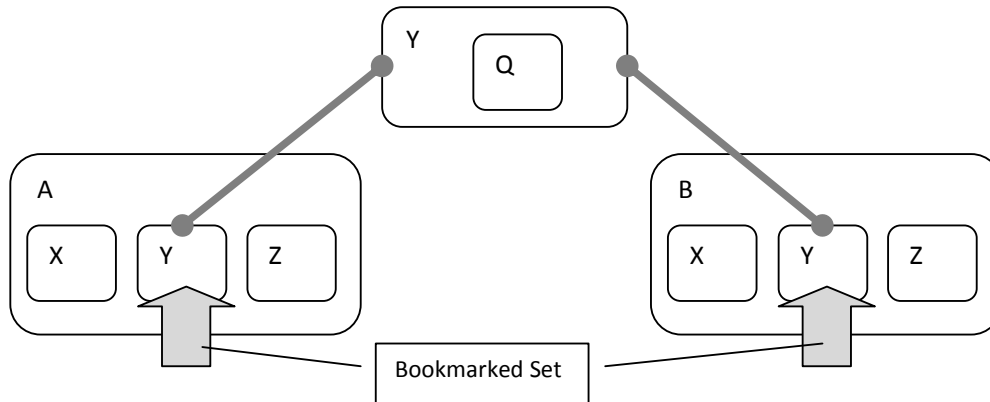


Figure 21, two different Bookmarked-Sets identify the same Set

In the above figure, Y can only be referenced in the Context of A and it is different from Y of B, even though Y itself is the same Set. This is the classic mathematical reference and exists in many variant forms, such as local or global variables in Object Orientated programming, shortcut's in Windows or the symbolic linking of files in Linux. Bookmarks capture this relationship by presenting a new Set parent that contains the same children, but the pointer (arrow in above diagram) that marks to the Set for which the operation will be applied is different.

Diagrammatically, Y of A is simply written as A and Y of B is written as B because the operator that will utilise the Set will automatically use the currently selected child. If no child exists, this produces an empty or null parameter. The specifics of how Sets are encoded may vary; one such example is covered in an expired patent *Computational Device for the Management of Sets*²⁵.

Bookmarked sets form the foundation pillar for the construction of Set structures that will house the TEE and enforce scoped protection.

²⁴ Peer Nodes are non-trusted in TAIS, even though they are chosen by the Node administrator

²⁵ <http://blueicon.com/uploads/Set%20Theory%20Device.pdf>

The TEE Knowledge Base

The TEE deals with changing accessibility of data, but does not allow data to be viewed, modified or created without introducing security risk. Information handling should be managed by the TAIS Zeta Environment (TZE) for measurable security.

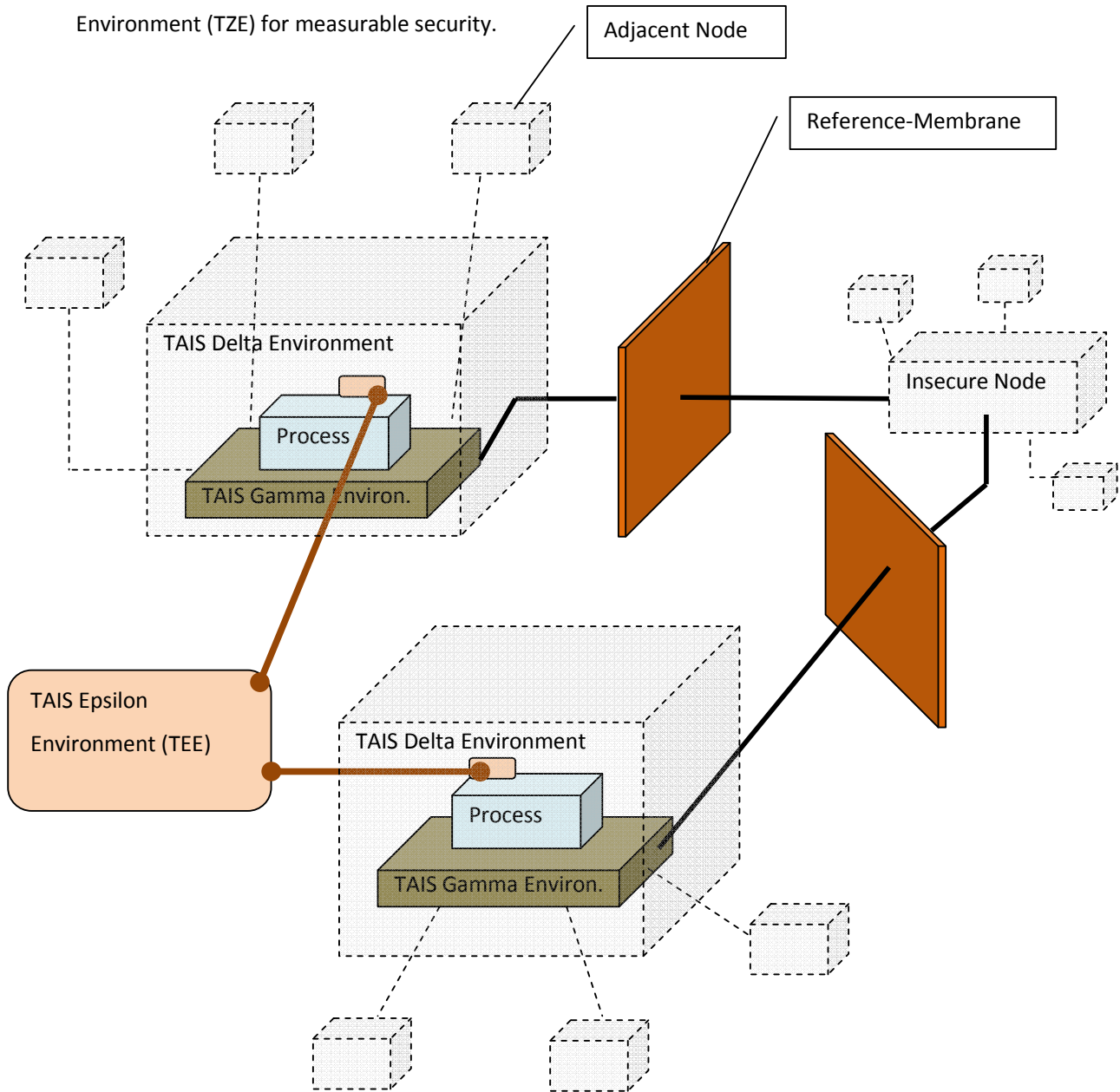


Figure 22, the TEE operates on top of a private network of TDE Nodes

Information inside the TEE is managed by the *TEE Knowledge Base* which holds the information to be secured. The TDE is already secured and the TEE extends this environment to store all private information to be managed by the system (a private network, see TDE). The TEE works naturally across a distributed base of TDE processes running concurrently that collectively store an unlimited amount of information, provided there is sufficient grid capacity.

Limiting information freedom with Infoscopes

Set Theory provides a data structure (the Set) with an infinitely configurable framework from which databases, filing systems, tree-controls, folders, Object Orientation or any other scoped protection mechanism can be derived.

A serious problem arises from this super-free structure. Sets allow any reference to be stored any number of times in any other Set. As a positive, it means that people can store information wherever they like in multiple places without concern that they may access the wrong version, because all *references* are active and up to date and are not just *copies*. As a negative, all this liberty means that it becomes difficult to remove a reference if it is no longer required. All references are made equal, so all identical references would need to be found and deleted at the same time in order to remove the item fully, in a similar way garbage collection works in Java and other emulated environments.

The Infoscope is engineered to cap this freedom and manage the total accessibility of the information unit. The Infoscope works in tandem with the Set's natural ability to store information and can act as a sophisticated filter to determine if information should become available for rendering.

The Infoscope uses 3 *Manifest-Criteria* to determine if information can be made available:

- **Information-Cleared:** The user has a reference to access the information
- **Mind-Cleared:** The user is allowed to access the information
- **Physically-Cleared:** The access device and access area are secured and the person is verified

The Infoscope model runs in stark contrast to the user/resource model²⁶ in general use today (2011).

Today, when a user is assigned access to a directory, then they 'have' the information and they share that access with every other user who 'has' access to that directory.

The Infoscope model is different and more empowering because it is **Information-Cleared**. Each user has access to their own individual files and cannot see each other's files. Some of these files will be shared references, so if either user that can access them makes a change, it will be visible (updated) on every other user's reference in their file collection. Once the user has access to information, they can forward it to anyone without introducing security risk. This is the property of **Mind-Cleared**.

Just stop there.

The most common information breach in the user/resource model occurs when authorised persons quietly forward information to unauthorised persons in a manner that goes undetected.

So, not only does the Infoscope solve this weakness with the user/resource model, but it's useful for people to be able to focus on structuring information in ways that make sense within the context of that information, instead of storing it in a manner that relates only to who can access it – if only for the reason that the security landscape is in constant flux as new people, policies and documents permeate the information system.

²⁶ Includes a family of models/systems where users or groups of users are given access to resources including hardware (like printers) and includes the sharing of common files

Since the Set Theory aspect of TEE storage allows information to be accessed in multiple places, it allows for information to exist in many information structures simultaneously. So the real power of the Infoscope comes to bear when information is heterogeneously mixed up and anyone can forward any collection of any size²⁷ to anyone simply because it's the most effective thing to do from a communication point of view and without consideration to its security.

The Infoscope model is a major shift in thinking because it allows a person to use information they were given by someone who was not authorised to use it.

The Infoscope model is a game-changer because it does not try to prevent information dissemination; it just prevents access to it.

The Infoscope, operating out of the TEE is a data access interface for multiple TAIS Zeta Environments (TZE) that allows users to view, create and modify information remotely at the edges of the network. The TZE ensures that each device and the route that connects that device meets the minimum TAIS risk profile, which is the property of **Physically-Cleared**.

²⁷ By storing information in the grid using the TDE, there is no limit to the size of any information structure and its transfer will always take finite number of Set operations instruction since no actual data is transferred.

Conceptually, it works like this:

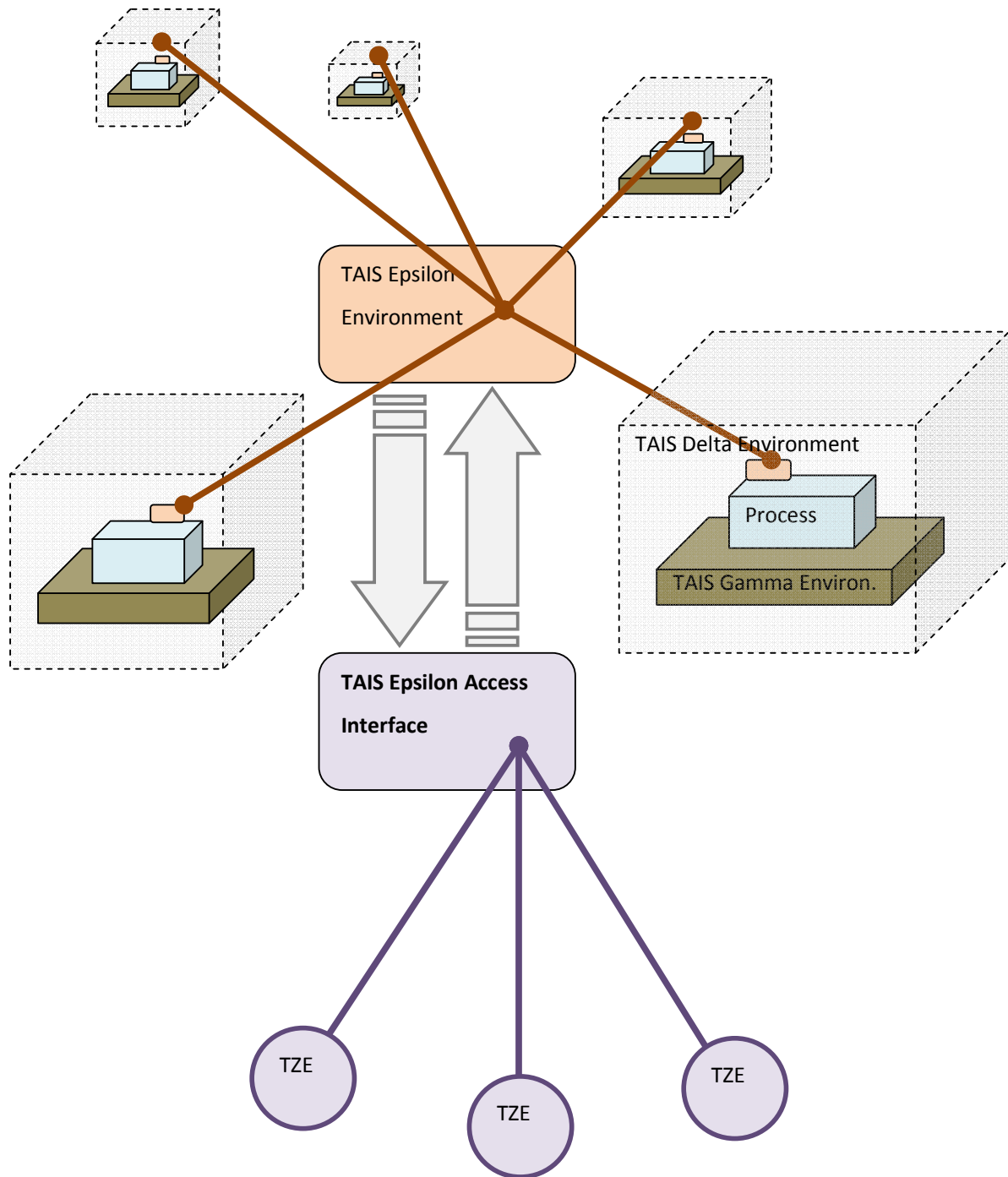


Figure 23, Multiple TEE (server) environments work in parallel to support every TZE client endpoint.

Physical control

One of the big differences (2011) TAIS presents is that it turns what has traditionally been a security specialist/analyst problem of securing information into a problem of ordinary physical security. In other words, if a nation state can physically lock down sufficient Nodes in a bunker with armed guards at the door, then the information can be kept safe. The converse is also a step forward, whereby a judge that orders the confiscation of key Nodes will allow authorities to gain access to that information, provided regulators insist on backdoor access for Nodes if they are physically compromised. Organisations specialising in communications (such as BlackBerry) are torn between providing private communication and adhering to wiretap legislation (2011).

Physical control means the information is stored within the secured TEE environment. It is held on multiple Nodes for redundancy and efficient peer transmission similar to a CDN network. When users access a unit of information, the TEE will be able to lock part of that information to avoid concurrent write errors. The actual information is made available to the user device once it has been verified that the device and route to that device are safe.

Logical control

Access to information is determined by the list of users that have access to the information contained in the Infoscope and the devices that are listed as acceptable on that Infoscope. TAIS is different from existing models (2011) that allow a single role (the administrator) to create users and determine which users can access what resources. Instead, TAIS recognises that the introduction of an additional privileged user (the administrator) creates its own security weakness as no information can be more secure than this weak link.

TAIS allows any user to create an Infoscope that is used to control the people and devices that can access the information that is contained within it. Information can only be stored in one Infoscope at a time, so information can only enter an Infoscope via 3 routes:

- Information is created inside the Infoscope
- Information is transferred from another Infoscope within the same TEE by the owner of the source Infoscope
- Information is transferred from an external TEE Infoscope

The Infoscope controls the accessibility entirely; the information can be 'dissolved' into the grid by removing all the users.

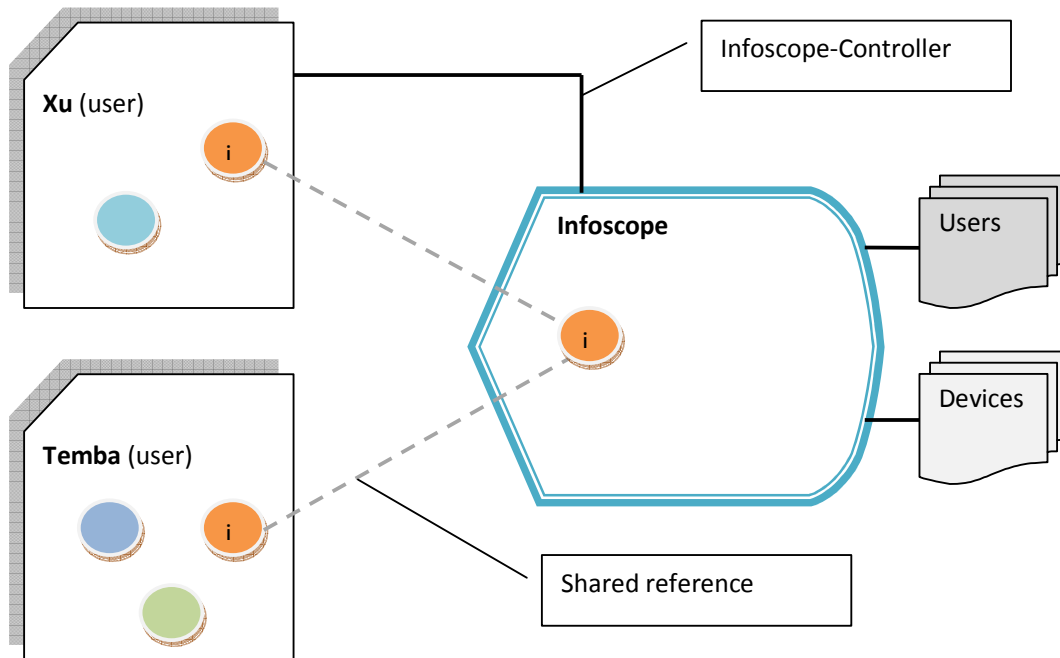


Figure 24, the Infoscope

Information can leave the Infoscope in 4 ways:

- Information is transferred to another Infoscope within the same TEE by the owner of the source Infoscope
- Information is transferred to an economic network (see TAIS Delta Environment)
- Information is destroyed
- Information is transferred to another TEE

Infoscope-to-Infoscope

In order to transfer information to another Infoscope, both owners of the source and destination need to authorise the transfer – consider the inverse case of a school information system that must prevent inappropriate information from entering the classroom Infoscope.

Infoscope-to-Economy

Once information migrates from a private form with controlled access into a public form it undergoes a permanent change. Economic networks (see TAIS Delta Environment) are expected (2011) to be capitalistic, but optimised for sharing in order to prevent patent and copyright issues. Once information is in the Economic network, it can be used without permission, but royalties are guaranteed. Therefore, the transfer can't be reversed as other people will have already invested their energy as they 'mash' the content with other content to produce new information.

Infoscope-to-null

Information can be deleted by the owner of the Infoscope without digital debris. Since both the physical and logical models dovetail to ensure there is only a single reference available for access control, once information is deleted it cannot be retrieved. The mathematical structure detailed in the next section below ensures that there is no record of the information being deleted – it simply disappears. Where auditing is required, customised Infoscopes must intentionally track everything required.

TEE-to-TEE

Since the TEE is an isolated private network, it can only control information that it contains and conversely cannot affect information managed by another TEE. TAIS makes provision for information to pass from one TEE to another as pragmatically, nation states may establish independent TEEs that they each trust.

Unlike the smooth management of information within the TEE, TEE-to-TEE transfers are expected to form the minority of transfers because users can only access information in their own TEE. This means that once information is transferred to another TEE, it is effectively inaccessible by all its users who could have otherwise belonged to multiple Infoscopes and would have been able to

continue accessing it. In practice, a copy may be made before transfer, but users will no longer be able to collaborate. Naturally, the ideal scenario is to operate out of a universal TEE.

The Infoscope dataflow

Since there are an almost unlimited number of ways of structuring the TEE, the Infoscope Set structure is provided here for illustration purposes only. The actual structure may differ as it represents a database that is driven by two interfaces.

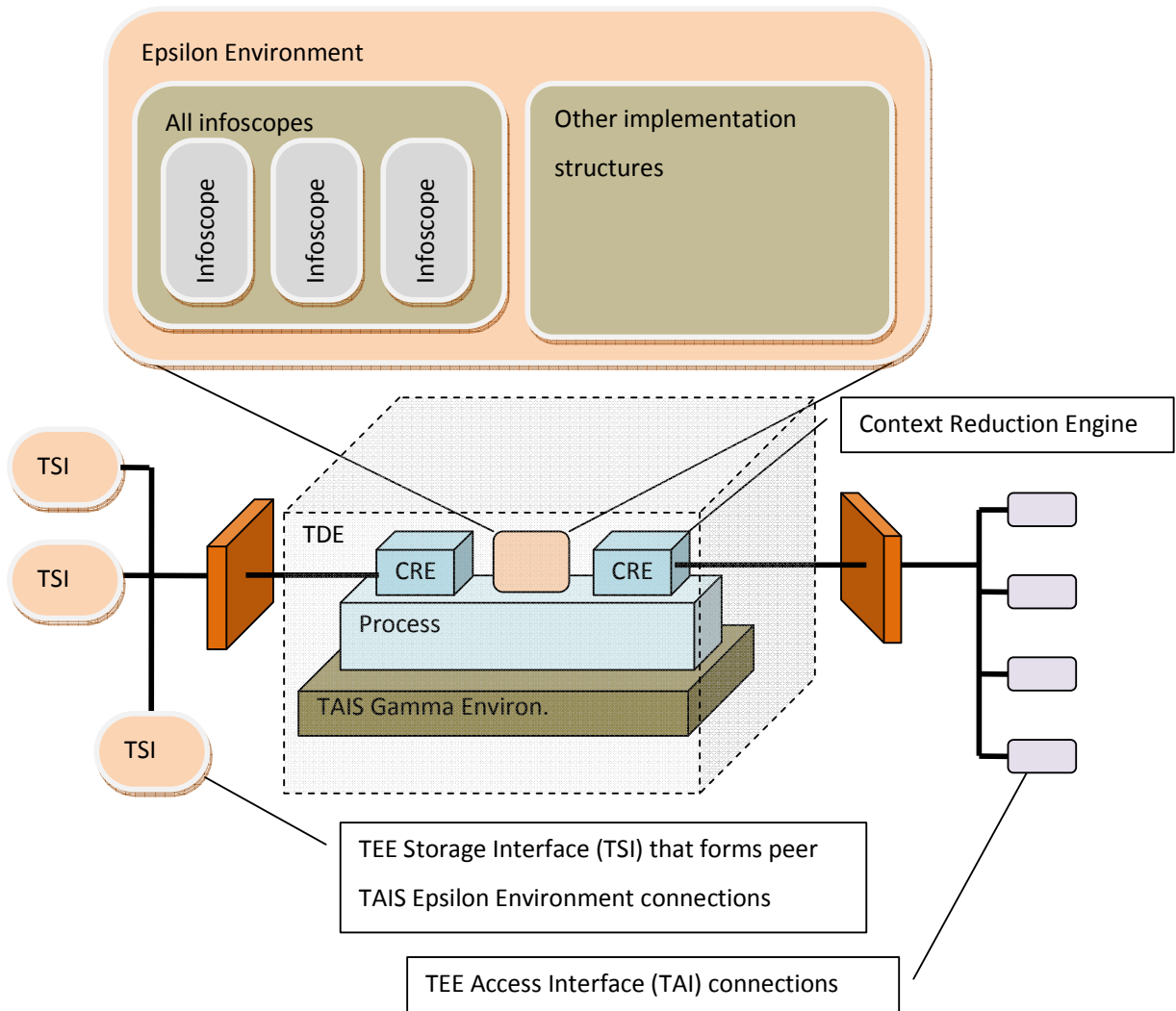


Figure 25, an abstract architecture for the TEE

All information storage maintenance is directed via the TEE Storage Interface (TSI) to peer while information manipulation and access is directed via the TEE Access Interface that links directly into the TAIS Zeta Environments (TZE) used at the edge points of the network to access information by users.

Criteria for abstraction

Any implementation is acceptable, provided the following conditions are met.

TEE Storage Interface

- Information locking and access remains consistent across a distributed data store
- The risk profile for every TEE storage process is equivalent

TEE Access Interface

- The device and route to the device are fully verified
- Access is only granted if both the device (and its route if listed) are present in the Infoscope
- The person accessing the device is authorised and listed on the Infoscope
- The Administrator and their access device is authorised before any Information-Unit (see figure below) is transferred into or out of the Infoscope
- The TAIS Zeta Environment should use an Entangled-Set Information-Unit itself as a reference to the information.

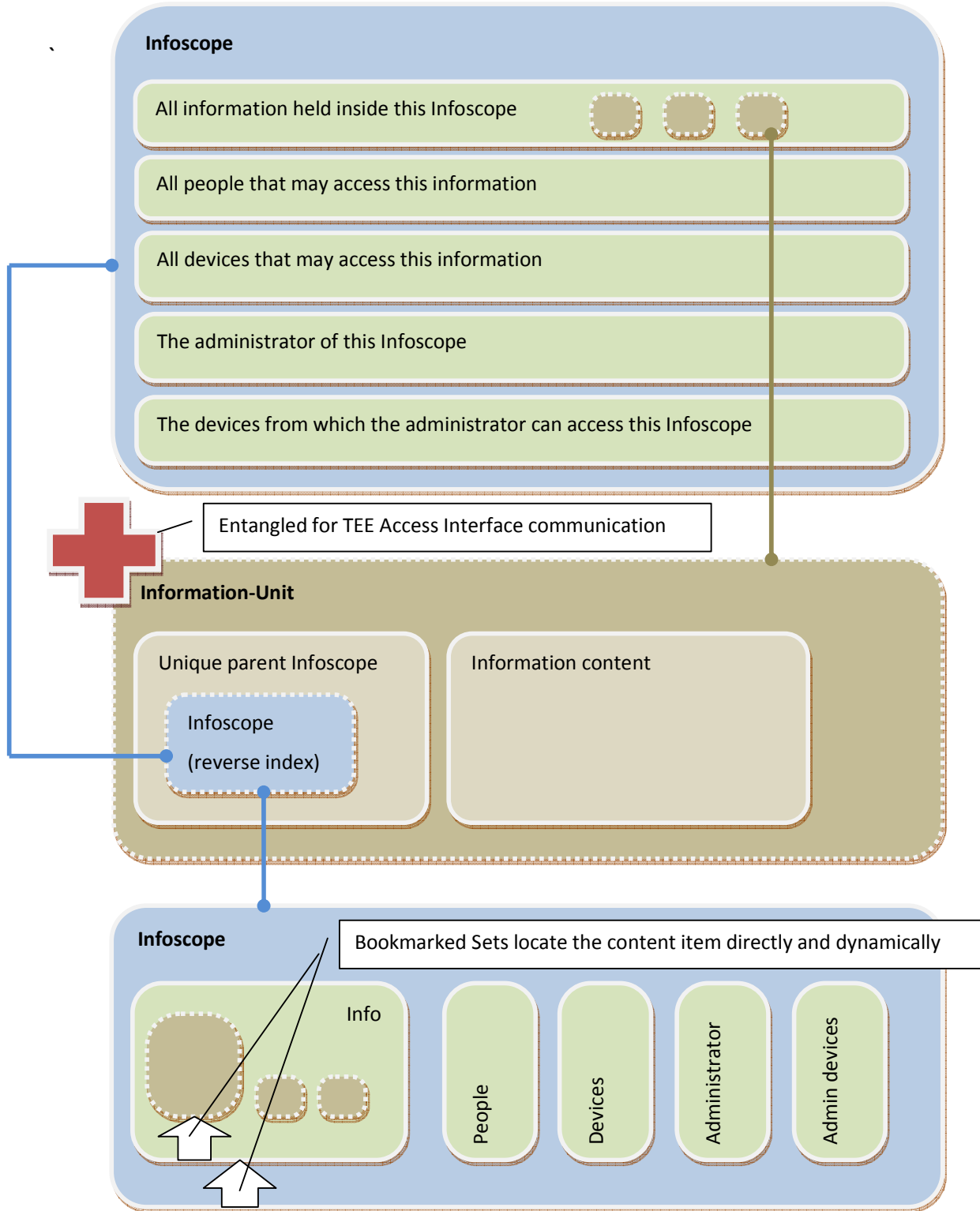


Figure 26, the Infoscope structure

Step 6

Prevent user access point leakage

ZETA ENVIRONMENT

The TAIS Zeta Environment (TZE) acts as an information render and capture device that resides at the edge of the network. The TZE is designed to operate in isolation from the TAIS Epsilon Environment (TEE) as a thin client that connects to the TEE server configuration via the TEE Access Interface.

The TZE is a critical element of security for the resulting information system, so much so that its integrity represents the *Physical-Security-Level* (PSL). The PSL is one of the 3 *Measurable-Elements* (covered in Step 7) that comprise the final security rating.

There are many possible architectures available for the TZE that could have a wide range of security ratings. This is possible because the TEE allows the Infoscope to specify which TZE endpoints are able to access the information inside it, so software engineers may choose to implement fast rendering TZE endpoints that avoid the cumbersome (but secure) configuration described next.

For high-end security deployments, it is necessary to utilise a fully secured TZE, one possible design is as such:

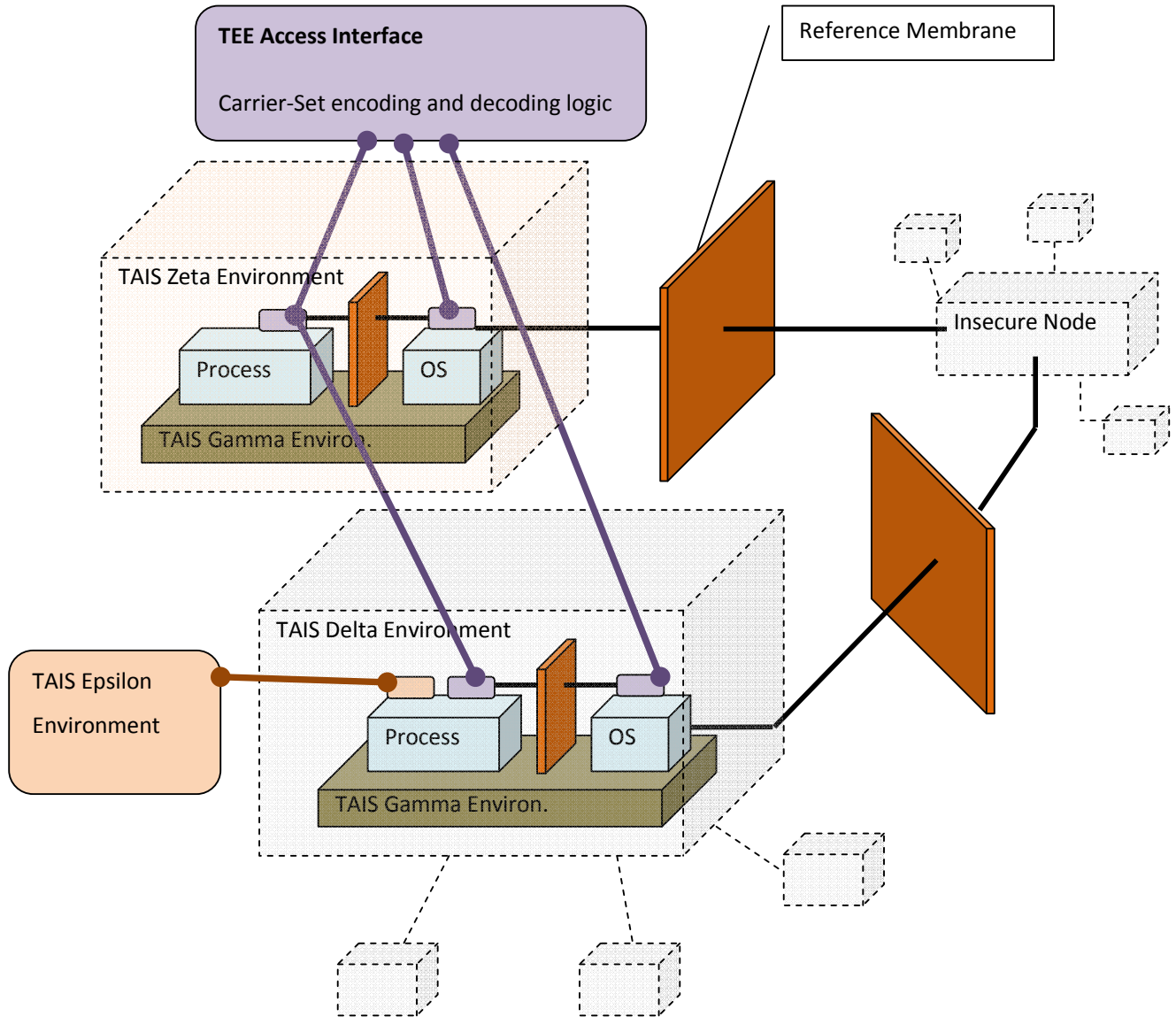


Figure 27, a sample TZE

The TEE Access Interface remains undefined within TAIS, however, all endpoints must utilise a consistent format in order to facilitate communication.

Step 7

Calculate and reintegrate the risk function

The 7th and final step for building an Information System with TAIS is to calculate the risk function for each *Measurable-Element* of the TAIS equation and drive accessibility to deliver the desired level of security. The *Measurable-Elements* of any Infoscope are:

Measurable-Element	ABR	Definition
Physical-Security-Level	PSL	The probability that a single TAIS Zeta Environment (TZE) access point will become compromised over its lifetime
Mind-Security-Level	MSL	The probability that an authorised individual will either deliberately, inadvertently or by way of coercion ²⁸ attempt to transfer knowledge out of the Infoscope using only their physical body and props external to the Infoscope during the time they have access to it
Information-Security-Level	ISL	The probability that any Node hosting the Infoscope becomes compromised

The total risk that a TAIS Information System presents is called the TAIS Security Risk (TSR) and is calculated as follows:

TSR = The highest probability of the following:

1. The highest Physical-Security-Level (PSL) of any TAIS Zeta Environment access point OR
2. The highest Mind-Security-Level (MSL) of any person with access to the Infoscope OR
3. The highest Information-Security-Level (ISL) for any Node hosting the Infoscope

²⁸ MSL can be further reduced if information is structured in the Infoscope in a manner that makes it difficult to reproduce externally from memory. For example, if access is deliberately fragmented so that no one person has access to all critical information in order to reassemble the data. Time trapdoor functions such as money transfers can also be mitigated within the content of the Infoscope, such as requiring multiple parties to simultaneously agree on an instruction.

Note that the ISR doesn't take into consideration the impact of an information breach because this is subjective and will be impossible to measure.

ISR simply means the chance of a single unit of information escaping the Infoscope and is an easy metric for non-technical people to engage.

Let's examine this measurement formula in slow motion.

Imagine that at some point in the not too distant future it is your role to secure your nation's most sensitive information. The president herself has ordered the government's information system to be rebuilt provided the designers can prove it to be 99.9997% secure for the nation's most sensitive information. As a result, you leave nothing to chance and choose a TAIS Information System because its deployment is mechanical and the theory has finally matured to the point where a proof of security has been established.

By consultation and agreement, your team divides the nation's information requirements into multiple Infoscopes for the various military and state agencies. Although you can easily create a new Infoscope, dissolve or merge Infoscopes at a later stage, your brief today requires that some of these Infoscopes operate at an incredible 0.0003% risk, meaning that the probability of a single breach during its life-time operation is at 0.0003%.

Even though you are capable of structuring all information at 0.0003%, you choose to increase the risk function for the less sensitive information because the deployment cost will be reduced. You will probably operate at around 1-5% of risk for about 95% of the nation's sensitive information.

Let's stay focused on the highest security requirement for 0.0003% risk, which lets say accounts for less than 1% of the nation's information store. Luckily for you, TAIS has been designed to be easy to deploy so as to reduce human error. The vast majority of the mechanical and technical aspects will have been address by the people who built and accredited the TAIS software and hardware. You are able to purchase one of these accredited TAIS systems that presents an Information-Security-Level (ISL) risk of 0.0002% and a Physical-Security-Level (PSL) of 0.0002% (but this doesn't include the risk if the hardware is accessible to the attacker and nor does it include the risk of electromagnetic radiation from TEMPEST scanners). This 0.0002% of conditional PSL risk means that you can crank out better security at a later stage without switching providers.

By using TAIS, your job is actually technical quite easy. You've already got an ISL of 0.0002%, so all you need is a matching MSL and PSL of 0.0003% to complete your brief.

Your first challenge is to deliver a 0.0003% PSL by ensuring no single TAIS Zeta Environment (TZE) access point can be violated. You do this by deploying every TZE access point with the preset factory conditional rating of 0.0002% within an isolated Faraday Cage to prevent electromagnetic leakage and equip it with high-end biometric authorisation . With the help of your accreditation partners, you evaluate this additional risk to be 0.0001%, which conveniently takes your true PSL to 0.0003%.

Finally, you have to address the human factor and ensure the MSL goes no higher than 0.0003%. To achieve this, you will have to carefully select (and reduce) the number of people who can access this environment and ensure they can be trusted. You may wish evaluate how vulnerable these people are from the threat of hostile coercion and keep their access identities a secret. Once your accreditation partners have evaluated this risk at less than 0.0003%, you can focus on the less critical information where your biggest challenge is now to ensure your spend is under budget.