An Overview of the Annex System

D.A. Grove T.C. Murray C.A. Owen C.J. North J.A. Jones M.R. Beaumont B.D. Hopkins
Defence Science & Technology Organisation
PO Box 1500, Edinburgh, South Australia 5111

Abstract

This paper describes the security and network architecture of the Annex system, a family of technologies for secure and pervasive communication and information processing that we have developed at the Australian Government's Defence Science and Technology Organisation. Our security architecture is built on top of a distributed object-capability system, which we believe provides an ideal platform for developing very high assurance devices. Our network architecture revolves around next generation networking technologies, including Mobile IPv6 and 802.11i wireless networking, but includes a small number of important extensions to improve security, robustness and mobility in the military context. A particular and unique contribution of our work is the tight integration of our very strong security architecture with next generation networking technologies. To complete the paper we describe our reference implementation of the Annex security and networking architecture, which consists of a number of devices known collectively as the Annex Ensemble.

1 Introduction

There is growing momentum within the Australian Defence Force (ADF) and its coalition partners to develop operations based on Network Centric Warfare (NCW) [2]. The goal of NCW is to simplify and unify the mechanisms for assured information sharing between interested parties in order to achieve a more efficient and effective fighting force. This is a particularly difficult problem for military communication and information processing systems because classified information at different security levels must be strictly partitioned. This is almost universally enforced by replicating, in isolation, computing and network resources for each security level. Clearly this does not scale and it will not allow the benefits of NCW to be fully realised. The solution to this problem is the holy grail of military computer security, to develop trusted systems that are capable of enforcing Multi Level Security (MLS) [24].

NCW will include the networking of assets such as fixed bases, mobile platforms, sensors, weapons systems and the individual war fighter into a Global Information Grid (GIG) [1]. To be truly effective, this ubiquitous GIG will need to be globally scalable, interoperable, reliable and secure. This ambitious vision will only be fulfilled in concert with the provision of a next-generation data communication network: one that can be easily, dynamically and automatically configured to provide seamless and secure network access to forces anywhere and at any time.

The Command, Control, Communications and Intelligence Division in the Defence Science and Technology Organisation is developing an experimental network, called Annex, which aims to provide enhanced connectivity between military forces as well as seamless connectivity to civilian entities using standard Internet Protocols. Annex is based on IPv6 [7] with Hierarchical Mobile IPv6 extensions [14, 25]. This allows for authentication and encryption using IPsec [13], provides flow markers to assist in the efficient management of streaming data, and supports addressing for mobile nodes. While maintaining full backward compatibility, Annex enabled systems are also enhanced with network awareness to improve real-time performance, robustness and security of communication between Annex enabled devices.

Although we have used Commercial Off The Shelf (COTS) technologies like those above wherever possible, the most fundamental design and implementation rule that we have applied during our development process is that very strong security mechanisms must be able to protect all aspects of the system. We believe that without security as the foundation, all other technology above it, no matter how sophisticated, is severely limited in its applicability to NCW. Indeed, we also feel that otherwise good technologies will eventually fail in much of the commercial marketplace if security is not equally well addressed.

The Annex security architecture is a foundation for providing very strong security using a universal system for access control based on *capabilities* [8]. Our capability system is implemented with trusted hardware and software that was developed entirely in-house, and together these form our Trusted Computing Base (TCB). The TCB allows Annex equipment to be used by any number of mutually suspicious, autonomous participants, with differing security policies and interests, facilitating secure access to and sharing of networked resources within the GIG. The Annex security architecture has been designed to naturally support the principles of *least authority*, *mutual suspicion* and *need to know* (see [5] for a glossary explaining terms) while allowing arbitrary security policies to be specified and provably enforced. Although minimal formal analysis has been performed on our TCB to date, we recognise that this will be essential in taking our devices beyond the proof-of-concept stage.

2 The Annex Security Architecture

The Annex security architecture provides an NCW platform that can support autonomous, mutually suspicious organisations to operate in coalition. Each organisation may define their own security policies regarding the sharing of resources and information, which the platform will enforce. The semantics of the underlying security model are encapsulated in a few simple rules. Hence, the model and its implementation lend themselves to formal analysis [27], which we believe will allow Annex to provide very strong security guarantees.

2.1 Capability Architectures

The Annex security architecture is based on a distributed capability system. *Capabilities* are unforgeable, authority-carrying references [8], although they may be copied under certain, strictly controlled conditions. They combine into one atomic entity the *name* of the object to which they refer as well as the *permission* required to access that object. This bond provides a very strong tool for avoiding the *confused deputy problem* [12] because it leaves no room for ambiguity about *what* authority is being wielded during any request to access any resource. Capabilities also simplify the implementation of trusted systems because they unify addressing and protection mechanisms.

Access requests in a capability system can only be authorised by capability *presentation*. The entire authority of a subject is defined by the capabilities it possesses and the transitive closure of what the possession of those capabilities permits. Subjects have no *ambient authority*, so a subject's authority can be limited by simply restricting the capabilities it holds. Therefore capabilities naturally promote *least privilege* [22] and the related principle of *least authority* [18]. The second principle is an extension of the first and includes causal effects that may arise from a subject exercising their permissions, including collaboration with other subjects.

Another important tool for enabling least authority is *delegation* [28]. During collaboration it is often necessary for one subject to wield part of another's authority. This can be achieved most easily if the second subject can delegate the appropriate authority to the first. In capability systems delegation is as simple as passing on a capability.

In spite of their benefits, capability systems have been criticised for the complexity required to manage capability distribution and revocation. Another perceived weakness of capability systems is a perceived lack of support for mandatory security policies [26]. Recent research is beginning to show, however, that these concerns may not be too burdensome if an appropriate capability model is used [18, 20, 23].

2.2 The Object-Capability Model

The object-capability model [18] manages complexity by providing natural support for abstractions. This allows fine-grained security policies to be specified using programming abstractions that "automatically" manage complexity in the same way that usual programming abstractions allow the construction of very complex software. Other abstractions allow mandatory security policies to be implemented in the object-capability model [18].

In an object-capability system capabilities address objects and allow method invocation, the only action that can be performed. Method invocation with object-capabilities is therefore similar to using protected procedures in CAP [29] or Hydra [15], but it is more securable by virtue of being the only available calling convention and also more flexible and fine-grained. Abstractions are implemented by objects whose methods expose functionality but hide implementation details, like in object-oriented programming. Abstractions may be arbitrarily composed because all object references (capabilities) are treated in exactly the same way.

The fundamental rule of object-capability systems is that "only connectivity begets connectivity" [17]. In practice, this leads to a small set of rules regarding capability propagation. These rules completely define the semantics of an object-capability system and provide a sound model on which to reason about the flow of authority and information.

2.3 Annex Capabilities

The Annex security architecture uses the object-capability model, but where capabilities are implemented using a *password capability* system [3] that has also been extended with *partitioning*. Annex capabilities are therefore stored as regular data using a password capability scheme, but may only reside within the protective bounds of the kernel. Outside of the kernel, objects reference capabilities using handles. When an object receives a new capability, for example during a method invocation, the kernel automatically replaces the capability with a per-object handle to that capability and creates a mapping between the two. Partitioning is completely transparent to objects, allowing them to treat handles as if they were real capabilities, but allows the enforcement of certain mandatory security properties including confinement, the simple security property and the *-property [19] because capability propagation can be enforced in accordance with the rules of the object-capability model.

Password capabilities are unforgeable because they contain a long and probabilistically unguessable password. Password capabilities also provide a universal capability representation that works across machine boundaries, taking advantage of Annex's network addressing architecture (see Section 3). A single Annex capability uniquely identifies a single object within the entire Annex network and allows the holder to invoke a subset of the object's methods. Table 1 depicts an Annex capability.

Table 1. Format of an Annex Capability

`DeviceID`	`ObjectID`	`CapID`	`Password`
64	48	16	256

The `DeviceID` field uniquely identifies the device on which an addressed object resides. This field corresponds to the least significant 64-bits of the device's IPv6 address and is known as its *host identifier*. This information is sufficient for routing invocation requests to a target device (see Section 3.4). The `ObjectID` field uniquely identifies a particular object on the target device. Each object is assigned a device-wide unique identifier. A type-marker, currently based on a cryptographic hash of the object's source code implementation, is also embedded within the `ObjectID` field to ensure that any confusion regarding an object's type will result in an invalid object reference. The `CapID` field is an object-wide unique identifier that is assigned when a capability is created. It uniquely identifies a particular capability from any other capabilities to the same object but with different permissions.

The `Password` field gives the capability its authority. While passwords provide no extra protection in the case that the underlying axioms of the system remain intact – the kernel will already be enforcing connectivity begets connectivity – they do provide a significant extra barrier against capability forgery on remote devices or via attacks on the kernel itself. Hence, a capability structure without a valid password is considered invalid and presentation will not provide the holder with any information, including whether the target object exists. Capability passwords are assigned randomly from a very large space in order to ensure that guessing a valid capability is infeasible.

2.3.1 Methods and Permissions

Methods are grouped into two classes. *User* methods must be defined separately for each object. They expose the interface of the abstraction that the object implements. *System* methods are inherited by all objects and provide the basic capability operations required of all objects. These operations include the *derivation* [3] of less powerful capabilities from those that are more powerful, and the *destruction* of individual capabilities.

A *catalogue* maps valid capabilities to a pair of permission bit-vectors: (`SystemPerms`, `UserPerms`). During compilation, individual methods are automatically mapped to individual bits within the appropriate bit-vector. Annex uses permissions to explicitly control the methods that may be executed by a particular capability, rather than the standard object-capability notion of a *facet* [17]. This avoids the need to instantiate multiple objects, which is advantageous on low resource embedded devices like those used in Annex (see Section 4).

2.4 Annex Capability Kernel

We have developed an operating system kernel that implements the Annex security architecture. The kernel is responsible for managing objects, overseeing the invocation of methods on those objects and for providing network transparency. Because our Annex Capability Kernel runs on trusted hardware (described in Section 4), Annex provides a complete platform for creating secure, object-capability based application software. Currently, however, it does rely on a small amount of untrusted hardware and software to provide network transport and untrusted user interaction services – but we are already working on extensions to our system to remove these limitations. In the mean time, our first generation Annex Kernels reside on secure PCMCIA cards that are hosted on untrusted commodity computers, such as COTS servers or hand-held PDAs.

The untrusted host may make method calls on objects to which it holds capabilities; however, the host only ever holds very weak capabilities and never holds capabilities to remote objects. The method calls available to the host mirror the user interface provided to the user, enabling untrusted user interaction via the untrusted host. The host is also responsible for forwarding remote method invocations over the Annex network. In this case, all remote invocations are encrypted so that only the remote destination kernel can read their contents, limiting the possible attacks available to the surrogate host or intermediate network to Denial of Service (DoS) and traffic analysis.

As shown in Figure 1 the kernel consists of four key components, along with a small set of trusted objects to expose core kernel functionality and hardware abstractions. The arrows in the diagram indicate that trusted applications

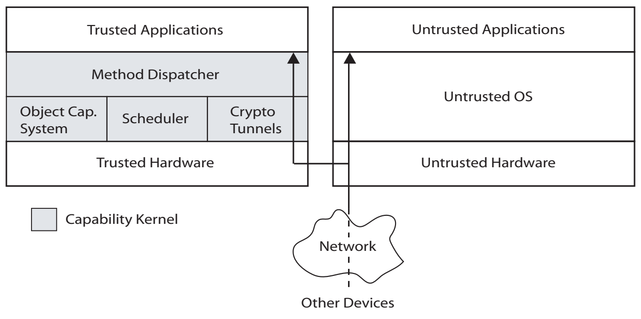

Figure 1. Software architecture of the first generation Annex Capability Kernel and its relationship to its untrusted surrogate platform.

may communicate with untrusted applications on their surrogate platform, or also with trusted applications on a different host via a secure tunnelling mechanism.

Firstly, a *scheduler* allows for multiple method invocations to be serviced in parallel. This allows the kernel to efficiently process blocking operations like network communication. All methods execute in separate threads that may only access the target object and the method invocation parameters. Although our prototype devices do not strictly enforce this isolation, we are developing hardware and software features that will guarantee absolute separation in the future. Our first generation scheduler is also non-preemptive so co-operative multitasking is required, which causes problems when using synchronous method calls, but we are updating our programming model and scheduler to overcome these limitations [11].

Secondly, a *tunnels* manager provides for mutually authenticated, confidential communication between object-capability kernels running on different devices. The Diffie-Hellman Station-to-Station protocol using Elliptic-Curve Cryptography (ECC) provides mutual authentication and symmetric-key agreement. The Advanced Encryption Standard (AES) operating in CFB mode ensures confidentiality. Tunnels can be cached for efficiency and different channels of communication can be multiplexed over one tunnel.

Tunnels are directly connected into the third component, the *dispatcher*. The dispatcher provides both network transparent method invocation and enforces capability-based access control to local objects. This centralised checking mechanism makes it possible to avoid error or abuse during capability verification. Furthermore, because Annex capabilities are the exclusive object naming mechanism the dispatcher provides mandatory access control that cannot be bypassed.

To achieve this the dispatcher relies on a number of components that make up the object-capability implementation. Our object-capability implementation comprises an *object-type database*, an *object store* and a password capability *catalogue*. Before a method invocation is dispatched the validity of the capability is checked with the catalogue, comparing the permissions it carries against those required by the method in the object's type definition. If they match then the object is loaded from the store and the method is executed.

Finally, our development environment includes tools for writing and compiling object type definitions. Each definition must include the object's method code as well as a description of the internal state that the object maintains, which is saved in the object store. Our initial implementation relied on static compile-time definitions of all possible object types, but we have now prototyped a loader that allows new object types to be instantiated at run time. This enables packaged collections of object-type definitions to be dynamically replaced. Thus the Annex architecture provides not only an access control framework but the means by which to develop and deploy arbitrary services to extend functionality.

2.5 Distribution and Revocation

The Annex security architecture is flexible enough to support a wide range of capability distribution and revocation schemes, implemented on top of the object-capability model. Possibilities for capability distribution mechanisms include centralised authority hierarchies and decentralised "webs of trust" as in PGP [30]. Our current proof-of-concept services, however, only use simple distribution schemes in order to allow us to experiment with the basic ideas before constructing more sophisticated mechanisms. It should be noted that once implemented, any mechanism should be readily reusable so long as it treats all capabilities it distributes identically.

Once a capability has been distributed, the only way to cancel its authority is by revoking that capability. While it is simple to revoke a capability by removing its record from the appropriate catalogue, determining which capabilities require revocation is very difficult without information regarding the propagation of capabilities between devices, otherwise it is impossible to know who holds any particular capability. We have developed a kernel-based algorithm for recording this information in a capability propagation graph, similar to that invented by Gligor [9]. Gligor's system, however, was only designed for single host systems, where the capability propagation graph is completely available and always correct. In contrast, the distributed and mutually suspicious nature of our system makes storing a global propagation graph impractical and imprudent. Hence, our revocation system was designed so that only local capability propagation information is stored by any one device. All devices cooperate to process this information in a distributed manner when revocation operations must be performed. In this respect our revocation algorithm is also similar to Miller's membrane pattern [16], but where our capability tracking is performed by the kernel rather than by the objects themselves. This allows our system to enforce transitive revocation policies, although it does slightly increase kernel complexity.

Since capabilities are partitioned so that objects only ever know about handles to capabilities, the dispatcher can always unambiguously identify capability transfers: they only occur as strongly typed parameters during method calls. When the dispatcher intercepts a capability transfer, it logs a (sequence number, capability, source object, destination object) tuple to an ordered list, called the *capability transfer table*. Entries in the table must be retained so long as the objects they point to exist, but old entries could be migrated to specialised log servers to reduce storage requirements on end-user devices. Furthermore, because maintaining and processing capability propagation information is expensive, our revocation system can be selectively enabled or disabled on a per-capability basis, as deemed appropriate by the issuer of a capability.

If it is later determined that a capability should be revoked – for example when someone no longer needs to perform a particular role, when someone no longer needs access to a certain resource, when it is discovered that an object was buggy or has been compromised, or when a device may have been lost or compromised – then the issuing device will be notified, which will remove the capability from its catalogue. It will also initiate distributed flushing procedures that recurse through the capability propagation graph of the affected capability, purging any of the now useless copies of the capability from other devices.

3 The Annex Network

The Annex network architecture reflects our vision of what an ideal communication network for NCW, unencumbered by current practice and technology limitations, will look like. This is not to say that the Annex network architecture is radically different from existing network architectures, but it is certainly far more focused on the future. Rather than simply taking existing technologies (along with all of their limitations) and figuring out how they could be applied to our vision, we first developed our architecture and then looked for any existing or developing technologies that could fulfil any of our requirements; wherever these technologies are inadequate we have developed our own. Consequently many aspects of the Annex network architecture closely resemble *next* generation networking technologies that are under development, although we have made a small number of significant enhancements that aid transparent wide-area mobility, robustness and provide strong security guarantees.

3.1 Wireless Networks

Wireless networking is becoming increasingly common. This is partially because wireless networks are easier and often cheaper to deploy. More importantly, however, it is because people like to be untethered. The ability for people to access the information they require at any time and place will help them to get their jobs done faster and better than ever before. There are two key issues that need to be overcome, however, before wireless networks will live up to their potential: reliability and security.

While the reliability of current wireless network technologies is not perfect, the explosion in the number of wireless network deployments around the world shows that users are willing to live with occasional performance glitches in return for the freedom that wireless networks provide. Furthermore, the reliability of wireless networking products is improving with every generation of new technology.

Similarly, the mechanisms available for securing wireless networks are also improving. Although these have been woefully inadequate for many years, leaving wireless networks dangerously exposed to unauthorised use, denial-of-service, eavesdropping and impersonation, they have recently matured to the level where they may be adequate in many circumstances. Taking stock of the lessons learnt from the flawed design of WEP and the more recent design compromises made for WPA, and standing back from the management/deployment complexities of VPNs, the IEEE has recently ratified a new security standard for wireless networks, called 802.11i. The strongest form of protection in 802.11i is called the Robust Security Network (RSN).

RSN uses 802.1X for authentication, which specifies an authentication and authorisation mechanism for allowing or disallowing clients to access a wireless link. It uses a virtual port-based approach, where both clients and networks must prove their identity and authority to each other before further network access is granted. After authentication, confidentiality and integrity is provided by the Advanced Encryption Standard algorithm in Counter mode with Cipher block chaining Message authentication code (AES-CCMP), using a key size of 128, 192 or 256 bits.

A number of 802.11i products that utilise 802.1X with AES-CCMP have already been accredited under the US Government's Federal Information Processing Standard (FIPS) 140-2 for sensitive but unclassified use. Although Annex uses 802.11i to protect the wireless link layer, this is intended to guard against casual interference only; strong security guarantees are provided by the security architecture described in Section 2.

3.2 IPv6 Networks

Future military networks are envisaged to operate on a global scale with multiple levels of mobility between end user devices and the supporting network infrastructure. IPv6 [7], the evolutionary successor to IPv4 upon which the Internet currently depends, is rightly seen as a major enabler for NCW and the GIG. A key aspect of this is IPv6's very large address space, which will be required to support the large number of networks and individually addressable nodes that are envisaged in the GIG. On top of this, Mobile IPv6 (MIPv6) [14] offers transparent support for computers that may change their point of attachment to the network.

During the MIPv6 handover period, however, a significant number of packets being streamed between highly mobile nodes may arrive late or be lost altogether. Consequently the MIPv6 protocol is not particularly suitable for real-time data flows between mobile hosts, such as for mobile telephony. Hierarchical Mobile IPv6 (HMIPv6) [25] is able to ensure faster handovers, but it is limited to fixed infrastructure deployments with a fairly static routing hierarchy. While this may be sufficient for most civilian networks, it is too limiting for highly mobile and transient military networks. We therefore designed several backwards-compatible enhancements to HMIPv6 for Annex networks that remove these limitations, as described below.

3.3 Enhanced Addressing

An important IPv6 network allocation policy is that of *route aggregation*, where prefix allocations are hierarchically assigned so that the number of routes that must be remembered by the Internet's core routers is minimised. IPv6's improved routing performance comes at a cost, however, as route aggregation leads to sparse address allocation. Annex leverages this sparsity to encode topological network information in IPv6 addresses, which can enhance addressing efficiency even for highly mobile nodes and infrastructure [10].

Annex employs a partitioned address allocation policy. This is done so that nodes can be uniquely identified regardless of their point of attachment to the network, and to exploit the structured network topologies that tend to emerge in large organisations for the purpose of enhancing routing efficiency and functionality.

Table 2 shows the structure of an Annex device's address, with suggested bit allocations. The prefix field represents the fixed part of the IPv6 address space that will be supplied by an Internet allocation authority and represents the entire Annex domain. We have obtained a provider-independent IPv6 allocation of `2001:4418/32` from APNIC in which to host a trial version of the Annex network. The *family* field is used to partition the global Annex address space into a number of domains or high level networks, where each can maintain its own independent administrative control, routing, security, and service requirements.

Table 2. Addressing for Annex networks

Prefix	Family	$\mathrm{Brick}_{\mathrm{L2,L1,L0}}$			Reserved	DeviceID
32	8	8	8	8	16	48

Annex Bricks are network infrastructure devices that provide a range of network and security services. Each family assigns their own $\mathit{Brick}_{\mathrm{L}n}$ fields to oversee routing, service handling, and structure within their own networks. The example address partitioning scheme promotes a three level (i.e. fairly flat) routing hierarchy, although more levels could be used. These 3 levels could correspond to global coverage (such as satellite) routers at L2, capital platforms at L1 and tactical vehicles at L0.

The host part in the lower 64 bits of the IPv6 address (16 bits of which are currently reserved for future use) contains the unique identity of any entity within the Annex network. This address remains unchanged for the lifetime of the unit and is never re-used. We envisage that these addresses will be physically burnt into the memory of devices to guarantee that they can not be modified. In some cases it may even be desirable to burn an entire 128-bit IPv6 identifier into devices, for example where disposable units are used or where security policies dictate that a device must only ever be used on a particular, designated network.

3.4 Network Awareness

Because Annex devices can be uniquely distinguished by their globally unique 64 bit host part and their current family, any device can be addressed by *uttering* its <prefix, family, $\mathrm{brick}_{\mathrm{L}n}$, DeviceID> tuple, regardless of the value of the $\mathrm{Brick}_{\mathrm{L}n}$ field(s), which represents the device's current point of attachment to the network. In particular, however, the <prefix, family, ZERO>/64 part of any Annex family's network is required to act as the home network for that family. Hence, any device can be contacted at its home address by zeroising the $\mathrm{Brick}_{\mathrm{L}n}$ fields in the tuple described above. If a correspondent node or intermediate router has a better idea about the current location of a device, however (i.e. it has any $\mathrm{Brick}_{\mathrm{L}n}$ stored in its binding update tables) it can utter the tuple with that information to facilitate more direct routing.

Uttering an address tuple describes our special form of IPv6 header construction, where source and destination addresses are each stored twice within the IPv6 header. The first source/destination pair corresponds to the source/destination pair found in a normal IPv6 header, and should (initially) contain the sending device's care-of address and the recipient's care-of address if it is known, or home address (i.e. where Brick$_{Ln}$ is zeroised) if it is not. Unlike traditional [H]MIPv6, these addresses are mutable by intermediate routers. The second source/destination pair should be stored in an IPv6 Destination Option extension header, and should contain the sending device's home address and the recipient's home address. These addresses are immutable but are also, therefore, amenable to compression. Since home networks are always located where Brick fields are zero and because the DeviceID on a foreign network will almost always be the same as on the home network, the storage requirements for the second set of addresses can almost be eliminated; in Annex networks these storage requirements will actually be less than those of incurred by standard [H]MIPv6 [10]. When the packet is finally delivered to the destination device, the main IPv6 header's source/destination pair will be replaced with the copies in the extension header and passed up to higher level processing as though the communication occurred directly between the two device's home addresses. If IPSec authentication is being used, care must be taken to ensure that the Authentication Header is created and verified with the same sets of predictably mutable source/destination address pairs in the main header and extension header. In addition, where ICMP status messages are generated in response to an uttered packet (eg for IPv6 path MTU discovery), care must be taken to route that response back to the home source address stored in the packet's extension header.

Whereas normal HMIPv6 must encapsulate such communication in a new tunnel for every level of Mobility Access Point (MAP), Annex MIPv6 routing allows the source/address pair in the main IPv6 header to be modified at will by intermediate Annex-MAPs. The simplicity afforded by avoiding the need to process extension headers/tunnel packets at every MAP will allow Annex-MAP routers to employ specialised hardware to perform very fast rerouting of Annex MIPv6 packets according to information stored in the MAPs binding caches. In addition, avoiding the use of preconfigured tunnels between MAPs means the routing topology may be dynamically altered to improve performance, robustness, security and/or other network characteristics. Such network awareness is very important in the military context, where directed threats against a communications network can change rapidly.

In addition to utilising hierarchical IPv6 Annex-MAP functionality, Annex devices can directly address other devices within a given MAP's sub-tree without involving the destination device's home agent: a source device need simply utter a message addressed to the unique host part of another device and a specified MAP. If the uttered device is being actively managed by that MAP (either directly or indirectly via some hierarchy) the message will be routed appropriately, as all MAPs know how to route towards devices beneath them in the MAP hierarchy if they are bound. Alternatively, if it is not, an Annex-augmented MAP may decide, depending on policy and/or other information contained in the message's IPv6 header, to report failure or possibly zeroise (or broadcast on) one or more Brick$_{Ln}$ fields and forward the packet on to broaden the search space for the destination device.

This augmented MAP functionality described above will provide Annex with more robustness and power than standard HMIPv6. Firstly, it provides a mechanism for packets to be very efficiently routed within the Annex domain. Secondly, it will give Annex devices the ability to contact other devices if they are reachable, whether their home agents are available or not, which is useful in a battlefield situation where local network connectivity has been maintained but the connection to remote network elements has been severed.

Enhanced topology awareness and mobile routing is not the only capability provided by network awareness in the Annex system. Other functionality includes classification-based routing, traffic priority and preemption and location-based services. More information and examples of how Annex IPv6 extensions can support intelligent routing decisions can be found elsewhere [10].

4 The Annex Ensemble

The Annex *Ensemble*, shown in Figure 2 and described further below, provides a reference platform for experimentation and development based on the Annex security and network architectures described in the previous sections. It comprises a range of devices for secure communication and data processing, designed to provide the Annex user with a trusted, personal platform for applications such as classified audio/video conferencing and data processing. When used together, these devices form a personal area network, which connects the war fighter to the GIG in a NCW environment.

All Ensemble components are personalised for the particular individual to whom they are issued. At the time of manufacture, each device is provided with a unique identity that distinguishes it from every other device within the Annex network. Devices are never reissued and are designed to be disposable, in the sense that it is expected that a particular individual will be issued with many different devices over the course of their service, many of which will be replacements for previously issued devices.

With the exception of the COTS iPAQ, the design and implementation of hardware and software for all elements of the Ensemble was conducted entirely in-house. Conse-

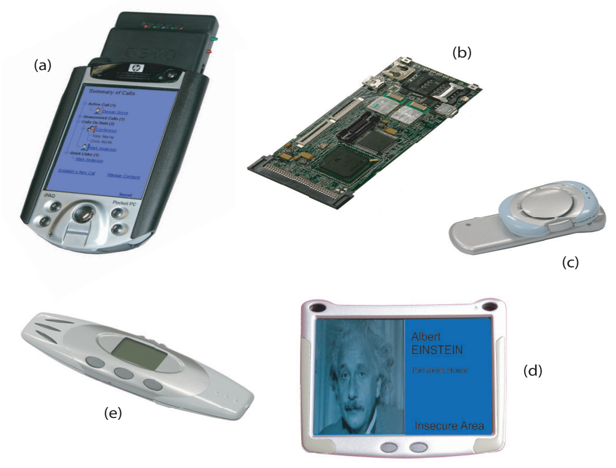

Figure 2. The first generation Annex Ensemble devices are: (a) MiniSec, which incorporates a (b) Secure Multi-function Card; (c) Button; (d) Badge; and (e) Codestick.

quently, everything except the iPAQ is considered "trusted". This is not to say that any of these components have been accredited to be secure yet – much work remains ahead of us down that path – but the Ensemble has allowed us to validate the basic ideas that underpin the Annex system through a process of rapid prototyping and experimentation. It has facilitated the incremental roll-out of Annex functionality, beginning with the end devices and moving towards the core network. For example, the Secure Multi-function Card which forms the Trusted Computing Base (TCB) of many of our first generation devices provides a reference implementation of the Annex Capability Kernel. By coupling this TCB with various COTS software and hardware components, we have been able to construct a number of functional Annex devices, including *MiniSec* and *Brick* prototypes. The MiniSec is an example of a typical Annex end user device, while Bricks are designed to form a core part of the Annex network infrastructure.

Our first generation Ensemble devices implement a number of prototype services, including secure telephony, video conferencing, network control, credential exchange and identity management. Some of these services rely on multiple cooperating Ensemble devices. For example, the MiniSec and *Button* operate in tandem to facilitate secure telephony, but including the *Badge* also enables support for video calls. We now describe each Ensemble component in more detail.

4.1 MiniSec

The MiniSec is a personal communication and data processing device that provides its user with secure access to Annex resources and applications. Our first generation MiniSec device consists of a COTS Hewlett Packard iPAQ hand-held computer augmented by a TCB in the form of the Secure Multi-function Card (described in the following section). The TCB is composed of custom hardware and software, implementing the Annex security architecture as described in Section 2. The TCB software includes the Annex Capability Kernel and application-specific objects. This software handles all sensitive data while the iPAQ, a potentially hostile environment, merely provides convenient network transport and GUI facilities for non-sensitive data or data that has already been encrypted by the TCB. Although our first generation MiniSec device therefore lacks MLS GUI facilities for showing sensitive information, we have already developed a second generation "MiniSec 2" device with full MLS display capabilities, which we hope to publish more details about soon. MiniSecs are connected into the wider Annex network using Mobile IPv6 over an 802.11i protected wireless LAN as described in Section 3.

Network transport and user interface software on the iPAQ interacts with the object system on the TCB through a user-space library and kernel driver. This allows software on the iPAQ to perform method invocations on (and receive callbacks from) trusted objects on the TCB. Method invocations are made using regular (albeit weakly permissive) capabilities that have been passed from the TCB to the iPAQ. Callbacks are delivered via a special trusted object that provides an interface mirroring callback requirements. Ideally the untrusted software on the iPAQ would not need to hold any capabilities at all, but it is required to support the user interface component of our first generation prototypes. Ultimately, however, we aim to implement user interface components with trusted software running on trusted hardware so that even weak capabilities do not need to be exported beyond the TCB.

We have prototyped a secure mobile telephony system based on the MiniSec platform, which allows a user to establish and manage any number of Multi Level Secure point-to-point or conference calls. Each distinct call or conference operates at a single, designated security level at any one point in time, although that level may be dynamically adjusted by the call's initiator. Any participants not cleared to a dynamically selected security level will be automatically put on hold until the conversation returns to their security level or below. Any user is free, however, to dynamically switch between any of the calls that they are involved in, or to create a new call, although only one call may be selected to be active at any one point in time. All call signalling and (unencrypted) audio data is managed by trusted hardware and software objects on the TCB, while the iPAQ simply transports TCB-encrypted data streams (using RTP over IPv6) and provides a convenient, although untrusted, user interface. Critical security information such as the classification level of the current call is displayed by several trusted LEDs on the TCB, however, so security breaches of the the untrusted software running on the iPAQ are limited to Denial of Service (DoS) attacks and traffic analysis. In our current implementation the untrusted nature of our GUI also makes it theoretically possible for the untrusted software to misdirect a call to an unintended recipient. The scope of this attack is greatly diminished by the trusted software, however, which still guarantees that both parties are appropriately cleared to talk to each other at the chosen security level. In addition we are working on another trusted output mechanism (via the Badge, which will be described shortly) that will display the remote party's identity and credentials to solve this problem more rigorously.

4.2 Secure Multi-function Card

Our Secure Multi-function Card (SMC) is a reconfigurable PCMCIA card that provides the trusted hardware platform for our TCB. It includes an ISO7810-compliant interface to the SMC card's host computer, a 32 bit ARM micro-controller for running the Annex Capability Kernel and trusted objects, a number of I/O facilities and other hardware resources, and a large Xilinx FPGA. The I/O and other hardware facilities include two Bluetooth devices, a SIM-card reader, a USB-host port, a Freescale MPC180 security coprocessor for accelerating Elliptic Curve Cryptography (ECC) key-agreement calculations, and several trusted push buttons and LED outputs. The FPGA is interposed between all of the other components and its job is to mediate all data flows between them, under the direction of a number of highly-trusted hardware abstraction objects running within the Annex Capability Kernel.

The FPGA can be reconfigured to suit different application requirements, but it is currently geared towards the encryption, decryption, routing and mixing of audio data to support our secure mobile telephony application. To this end the SMC provides a number of data channels, either input or output, that terminate on the PCMCIA host, or in one special case at a Bluetooth channel to the Button (which is described in the following section). Each of these channels are routed through a block-based, key-agile AES engine operating in CFB mode that encrypts data leaving the TCB and decrypts data entering it. Encrypted audio data sent to the PCMCIA host can then be transmitted to TCBs on other MiniSecs. Decrypted audio data can either be re-encrypted and sent to the Button for immediate playback, or first directed through a PCM mixing block in the case of a conference call. Channel routing is performed according to a switching matrix within the FPGA, which is in turn controlled by highly-trusted objects running within the Annex Capability Kernel.

4.3 Button

The Button is a wearable accessory that clips on to a shirt collar and provides a secure hands-free audio I/O device for the Annex Ensemble. In its current incarnation it provides 4 hours of talk time and significantly longer standby time. It captures audio in 64-kbps μ-law format using a sensitive electret microphone and outputs audio via a built-in speaker or headphone attachment. A dedicated acoustic echo cancelling chip allows the Button to operate in full-duplex speaker phone mode. The Button also incorporates a tri-colour LED that conveys MLS status to the user, and a push button for user action signalling, for example to accept or reject incoming calls.

Each Button is designated for use with a particular SMC card by Bluetooth pairing. The Button's I/O audio channels are connected, via the Bluetooth link, to I/O channels on the SMC so that audio recording and playback can be controlled and routed by trusted objects in the TCB. A Class 2 Bluetooth v1.2 link is used, which has a range of about 10 metres and minimises radio frequency interference between the Bluetooth and 802.11b devices on the MiniSec. Communication over the link is carried by a custom RFCOMM-based protocol, with all data encrypted by a FPGA-based 256 bit AES engine. Finite life time Key Material for the crypto engine is supplied by the SMC over a physical link once a day, for example during recharging.

4.4 Badge

Complementary to the Button, the Badge is a wearable display that provides secure display and video capture facilities for the Annex Ensemble. We have prototyped the Badge using an embedded Linux computer, coupled with an FPGA, a touch sensitive LCD, forward and backward facing cameras, video compression hardware and a Bluetooth radio. Like on the Button the FPGA encrypts and decrypts Bluetooth communications, but on the Badge it also provides digital signature and verification facilities. This allows the Badge to be used as an MLS data display, for example for classified imagery or video streams, or, potentially, as a trusted GUI for certain MiniSec applications.

The Badge also acts as a trusted, context-aware identity badge that alters its display depending on the environment, based on availability (or absence of) securely transmitted signals from nearby Bricks (see Section 4.6). Outside a Defence installation, for example, the display may become inactive, whereas inside a high security facility it will display the full credentials of its holder. The Badge can also be used to assist with the verification of an unknown party's credentials. For example, a security guard might use his Badge to display the identity photo of a visitor, sourced from their Codestick (described next) or a trusted database, thus protecting against the use of forged, stolen or tampered devices.

4.5 Codestick

The Codestick is a high assurance credential exchange device. Its primary function is to unburden Defence personnel from having to remember multiple passwords, carry around multiple security tokens or forward on security clearance details in advance of meetings. The Codestick is designed as a tamper-proof device that can only be activated by biometric authentication, which we hope will help it to achieve very high levels of security accreditation.

Two individuals with their own personalised Codesticks can exchange personal security credentials directly, without recourse to a third party. Secure cryptographic protocols, directed communication mechanisms and trusted high assurance design ensures that the credential exchanges are truly peer-to-peer and provide integrity and confidentiality for all data involved in the transactions. This is a substantial improvement over Defence's current credential exchange system, where each party must get their own security officer to manually exchange clearance information with the other party's security officer – an awkward, slow and error-prone process.

The Codestick also provides single sign-on functionality that allows its owner to automatically log on to Enterprise networks and any number of web-based systems, and digital signature, encryption, decryption and signature verification facilities to protect its owner's email.

Many functions of the Codestick have already reached stable operation. A user trial of the device is now being prepared. More information about the Codestick can also be found in patent number WO 2004/109973 [4].

4.6 Bricks

Although not strictly part of the Annex Ensemble, i.e. the devices that one would carry around on their person, Bricks are an essential component in the Annex system. Bricks are infrastructure devices that form the backbone of the Annex network or provide other Annex enabled services. In the same way that our MiniSec is comprised of a COTS iPAQ augmented with an SMC card, we have developed a number of prototype Bricks by adding an SMC card to existing server or network hardware. Coupling these SMCs with appropriate trusted object implementations makes it possible to draw these platforms under the control of the Annex capability framework, with the many authorisation and security related benefits that this affords.

The first Brick modules we implemented provide gateways between our secure VOIP system and the Public Switched Telephone Network (PSTN) and GSM phone networks. Calls from MiniSec devices to a public telephone system transit the secure VOIP network to a gateway point, which, after establishing a new, normal telephone call between the gateway device and destination telephone, then acts as the intermediary for audio data between both systems. We used OpenH323 [21] to implement the PSTN side of our gateway device. Outgoing calls can be made by a fixed line telephony card, or, using an extension to OpenH323 that we developed, via a mobile telephone supporting the Bluetooth Hands Free profile [6]. Incoming calls from either the PSTN or a Bluetooth enabled mobile phone are supported by the same processes in reverse.

We have also created experimental IPv6 routers and wireless access points whose operational parameters can be controlled by trusted objects running under an Annex Capability Kernel. We achieved this by connecting a serial line from a trusted "router control" object on a MiniSec device to the serial console (i.e. configuration) port on a Cisco router. As well as implementing the serial communication channel, the control object also implements a state machine that models the router's command line interface to abstract away the configuration terminal's stateful behaviour, thus exposing a simple but powerful interface for accessing the extensive range of configuration options available via the router's executive. We made use of this to implement a proof-of-concept Multi Level Secure network admission policy, such that only appropriately classified MiniSec devices could make use of a particular network.

Although the details of our implementation are beyond the scope of this paper, we used the following basic approach: (1) the router's initial traffic filtering policies completely firewall all newly connected clients from sending or receiving any traffic, other than to the MiniSec device controlling the router; (2) the router control object would cause the broadcast of an authentication beacon containing a weakly permissive capability for itself to any potential MiniSec clients that have roamed onto the network, thus allowing them to initiate capability-based authentication with the Brick; (3) this new client would address the router control object using the beacon capability and attach a capability to a credential object proving that it is cleared to the appropriate level (see [20] for more details about this step); and (4) the router control object would modify the router's traffic-filtering rules allowing the newly authenticated MiniSec device to access the network.

5 The Annex Testbed

The Annex testbed is a reference implementation of all of the Annex elements described in this paper. It includes several dozen complete ensembles, although most elements are only early prototypes. The Annex IPv6 network, which is distributed by a wide area ATM network connecting DSTO sites almost 1000km apart, is composed of a number of Cisco routers and 802.11b access points that employ 802.11i link-layer security; more recently we have also linked this network to the IPv6 Internet. Our MiniSecs use a modified Mobile IPv6 for Linux (MIPL) implementation, which allows their users to seamlessly roam with their enesmbles throughout the entire network. This provides an ideal platform for experimentation with the Annex framework and novel security and mobility applications.

We began conducting both on-site and off-site demonstrations (via the IPv6 Internet) of our secure mobile telephony application using the Annex testbed in August 2005. This environment allows users to make any number of concurrent point-to-point calls, conference calls, calls to external telephone numbers by the PSTN or GSM mobile phones, all under the secure control of the Annex capability system. Not only does this provide a robust platform for further experimentation and implementation, we also believe that it helps to demonstrate the viability of the overall Annex concept and the technologies that underpin it.

6 Conclusion and Further Work

This paper summarised the security and network architecture that underpins a family of DSTO developed technologies and devices that are collectively known as the Annex System. The centerpiece of all of our devices is the the Annex Security Architecture explained in Section 2, a trusted platform for distributed object-capability based software, which is inherently more secure and amenable to formal analysis than programs that are developed using current software engineering practices. In addition, Section 3 described how the distributed aspect of our security architecture is based squarely on next generation networking technologies, such as IPv6 and 802.11i wireless networking, but includes a small number of important extensions to improve security, robustness and mobility in the military context.

Although some researchers have been separately investigating similar security and network architectures, we are not aware of any other projects that have made significant progress – or even attempts – to combine these types of technology. Furthermore, because both of these concepts must be used in tandem to truly progress the state of the art in secure mobile devices, very few of these projects have been forced to go beyond simple proof of concept examples. In contrast, our contribution has been to not only combine advances in high assurance software and mobile devices, but also to demonstrate complex real world applications based on the fusion of these ideas.

In Section 4 we introduced our reference implementation of the Annex security and network architecture, a Multi Level Secure ensemble of devices including: the MiniSec, a wide area communications hub with an example MLS voice application; the Button, a small, secure Bluetooth based speakerphone; the Badge, a secure, wearable, context sensitive display; the Codestick, a high assurance credential exchange device; and Bricks, a range commodity network infrastructure devices retrofitted for subservience to an auxiliary, trusted Annex security device.

We have been performing demonstrations and conducting scientific trials of our ensemble of Annex devices since late 2005. Since then we have also been working on a second generation of Annex devices, including a far more refined version of the Codestick device and also a substantially improved MiniSec. Our "MiniSec2" device, which we hope to publish more details on soon, supports Multi Level Secure operation for *any* unmodified, off the shelf applications, including document editing and viewing, email, web browsing, voice and video communications and more.

References

[1] David Alberts and Richard Hayes. *Power to the Edge: Command and Control in the Information Age*. CCRP Publications, 2003.

[2] David. S. Alberts, John J. Gartska, and Frederick P Stein. *Network Centric Warfare: Developing and Leveraging Information Superiority*. CCRP Publications, 1999.

[3] M. Anderson, R. D. Pose, and C. S. Wallace. A password capability system. *The Computer Journal*, 29(1):1–8, 1986.

[4] Mark Anderson. Credential communication device, December 2004. International Publication Number WO 2004/109973.

[5] ATIS Committee T1A1. *ATIS Telecom Glossary 2000*, 2001. http://www.atis.org/tg2k/.

[6] Bluetooth Car Working Group. *Hands-Free Profile specification version 1.5*, 2005. http://www.bluetooth.com.

[7] S. Deering and R. R. Hinden. Internet Protocol, Version 6 (IPv6) (RFC 2460), December 1998.

[8] J. B. Dennis and E. C. Van Horn. Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9:143–154, 1966.

[9] Virgil D. Gligor. Review and revocation of access privileges distributed through capabilities. *IEEE Transactions on Software Engineering*, 5:575–586, 1979.

[10] D. A. Grove, M. Anderson, and C. J. North. Enhancing Hierarchical Mobile IPv6 addressing for the Annex architecture. In *Proceedings of the 7th International Conference on High Speed Networks and Multimedia Communications (LNCS 3079)*, pages 492–502, 2004.

[11] D. A. Grove, T. Newby, C.A. Owen, C.J. North, A.P. Murray, T.C. Murray, A.V. Uzanov, and T.J. Cuthbertson. The second generation Annex TCB. In *In preparation*, 2007.

[12] Norm Hardy. The confused deputy (or why capabilities might have been invented). *Operating Systems Review*, 22(4):36–38, October 1988.

[13] IETF IPsec working group. IPSec (RFC 4301-4309), December 1998.

[14] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6 (RFC 3775), June 2004.

[15] Henry M. Levy. *Capability-Based Computer Systems*. Digital Press, 1984.

[16] Mark S. Miller. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA, May 2006.

[17] Mark S. Miller, Chip Morningstar, and Bill Frantz. Capability-based financial instruments. In *Proceedings of Financial Cryptography 2000*, February 2000.

[18] Mark S. Miller and Jonathan S. Shapiro. Paradigm regained: Abstraction mechanisms for access control. In *8th Asian Computing Science Conference (ASIAN03)*, pages 224–242, December 2003.

[19] Mark S. Miller, Ka-Ping Yee, and Jonathan S. Shapiro. Capability myths demolished. Technical report, HP Laboratories, 2003.

[20] T.C. Murray and D.A. Grove. Non-delegatable authorities in capability systems. *Submitted to the Journal of Computer Security*, 2007.

[21] OpenH323 Project. `http://sourceforge.net/projects/openh323`.

[22] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1208–1308, September 1975.

[23] Jonathan S. Shapiro and Samuel Weber. Verifying the EROS confinement mechanism. In *Proceedings of the IEEE Symposium on Security and Privacy, 2000*, pages 166–176, 2000.

[24] Richard Smith. Multilevel security. In Hossein Bidgoli, editor, *Handbook of Information Security: Threats, Vulnerabilities, Prevention, Detection and Management*. Wiley, New York, 2005.

[25] H. Soliman, C. Castelluccia, and K. El Malki. Hierarchical Mobile IPv6 mobility management (hmipv6) (RFC 3775), August 2005.

[26] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Anderson, and J. Lepreau. The Flask security architecture: System support for diverse security policies. In *Proceedings of the Eighth USENIX Security Symposium*, pages 123–139, 1999.

[27] Fred Spiessens and Peter Van Roy. A practical formal model for safety analysis in capability-based systems. In *TGC 2005, LNCS 3705*, pages 248–278, 2005.

[28] Marc Stiegler. A picturebook of secure cooperation, 2004. `http://www.skyhunter.com/marcs/SecurityPictureBook.ppt`.

[29] M.V. Wilkes and R. M. Needham. *The Cambridge CAP Computer and its Operating System*. Elsevier North-Holland, 1979.

[30] P. Zimmerman. *Pretty Good Privacy*, June 1993. Distributed with PGP software.