The Pygar Project
for OpenBEDM

A Conceptual Architecture for a Secure Information Sharing System – Design Elements of OpenBEDM

pbaker@wwnsoftware.com

Release 1 - June 2012

Table of Contents

A Conceptual Architecture for a Secure Information Sharing System

Executive Summary

A responsible approach to information sharing should allow and encourage sharing while blocking arbitrary access to sensitive, private data. In addition, information sources are large operations, they are located at different sites, and they are administered separately. Effective information sharing must integrate the sharing operations while respecting the differences between the organizations responsible for the data sources. Above all, it is necessary to win the active cooperation of the sources because each must contribute to the overall success of the mission.

Key factors in eliciting cooperation are the reduction of risk to a minimal level and the assurance of mutual rewards from cooperation. These key are orthogonal to the goals of conventional information systems; therefore, it is technically possible to enhance conventional information systems by adding new components that enable cooperation around secure information sharing. Replacement of systems is impractical and unnecessary.

The distinctive feature of the secure sharing system architecture is the addition of a new element: a blind agent which is an independent system that reviews encrypted versions of information and finds encrypted connections between the elements. It can be compared to a federated data concept; but, the federated system introduces a central system that has access to all the data sources unencrypted, which is a major security risk. The blind agent cannot see unencrypted data and its findings remain encrypted in a manner that the blind agent cannot reverse. The value of the blind agent is that it can tell the information sources what they need to share to produce valuable results. Risk is minimized because only the necessary information leaves the protection of its archive in a sharing transaction. Rewards are ensured because transactions are recorded and available for audit.

Foreword

A complex system comprised of software components, hardware units and multiple users can be comprehended at a high level by studying its architecture. Architecture specifications require multiple viewpoints and must explain the style of system integration, the implementation choices and various distinguishing design motifs. In short, architecture requires a long narrative.

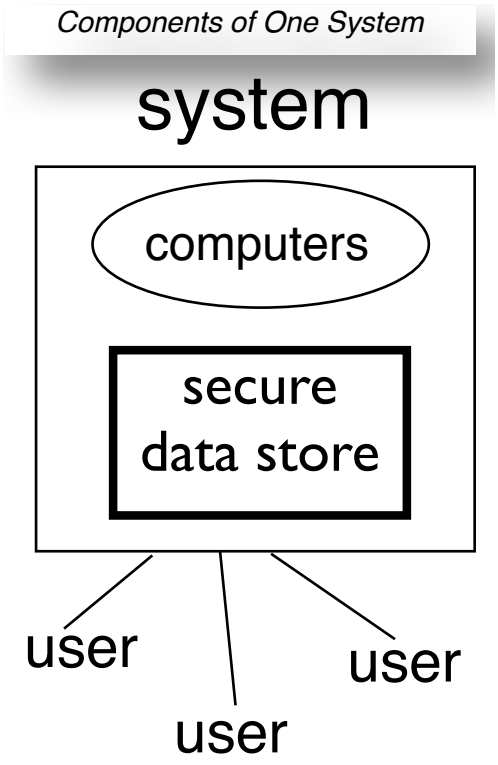
The subject of this document is a complex system for secure, responsible information sharing that follows the principles of blind encrypted data matching (BEDM), strict separation of encrypted data from encryption keys, and enhanced physical and network security. It is a feasible system because it does not replace existing systems. Rather, it is an evolutionary development from legacy operational systems. Evolutionary software is a response to a basic fact of life: if the current system is mission critical, it cannot be shutdown for replacement. New capability must be implemented on the current foundation. The new capabilities are built in place and to custom specifications but, fortunately for the budget, the capabilities are constructed from a set of reusable software components. Although the new software components are accompanied by documentation, component descriptions alone cannot convey the integrated operation of the distributed system. For that reason, this document provides an architecture description. It should be essential reading for anyone interested in secure information sharing.

The main obstacle to understanding a system architecture is that it is necessary to grasp several viewpoints which elucidate the system through their combination of insights. Reading and writing however are linear. This document considers a series of viewpoints and we recommend that the order should be followed on first reading because each section assumes a familiarity with the preceding section.

The Distributed Systems Viewpoint

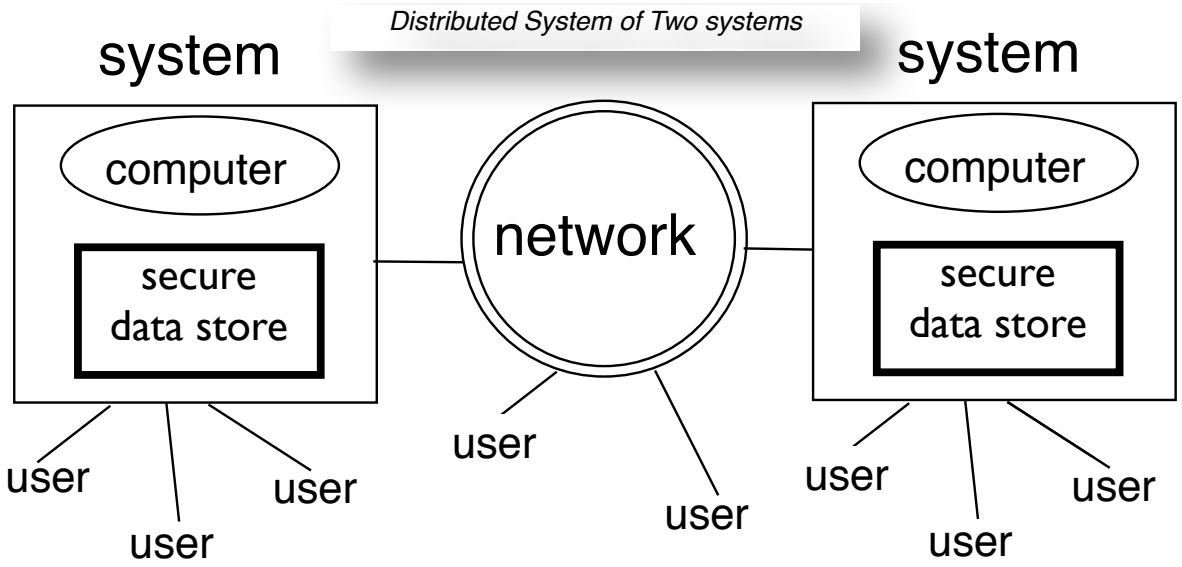
Standalone Systems

To begin, let us consider the elements of a single information system. An information system has three essential components. First a system has a computer component which may be a single computer or multiple computers. Second, it has a secure data store for sensitive data that must be protected because of privacy considerations and also because the betrayal of operational secrets can compromise the mission. Third, a system has users who represent the strength and the fatal weakness of any organization. Users get the job done and accomplish the mission but they also make mistakes in judgement or in execution of their responsibilities. Moreover, users are subject to outside pressure and may even be paid or persuaded to reveal secrets. Users are also known to be asleep on the job at a critical moment. Thus, most systems have active software processes that take over some responsibilities of the users. For the architecture of a secure information system, it doesn't make much difference whether the user is a person or a software process. Both need data, both have access rights, and both can misuse the access rights. So to simplify the discussion, we talk about users but mean people, analysis workbenches, and independent software processes.



From the security standpoint, we have just omitted the most important part of a secure system: management - the administration of facilities, the formulation of policies, and the authorization of users. Without sound management, information security cannot exist. We assume each system has sound management and we deal then with the new problems that are unique to multiple systems.

Distributed Systems

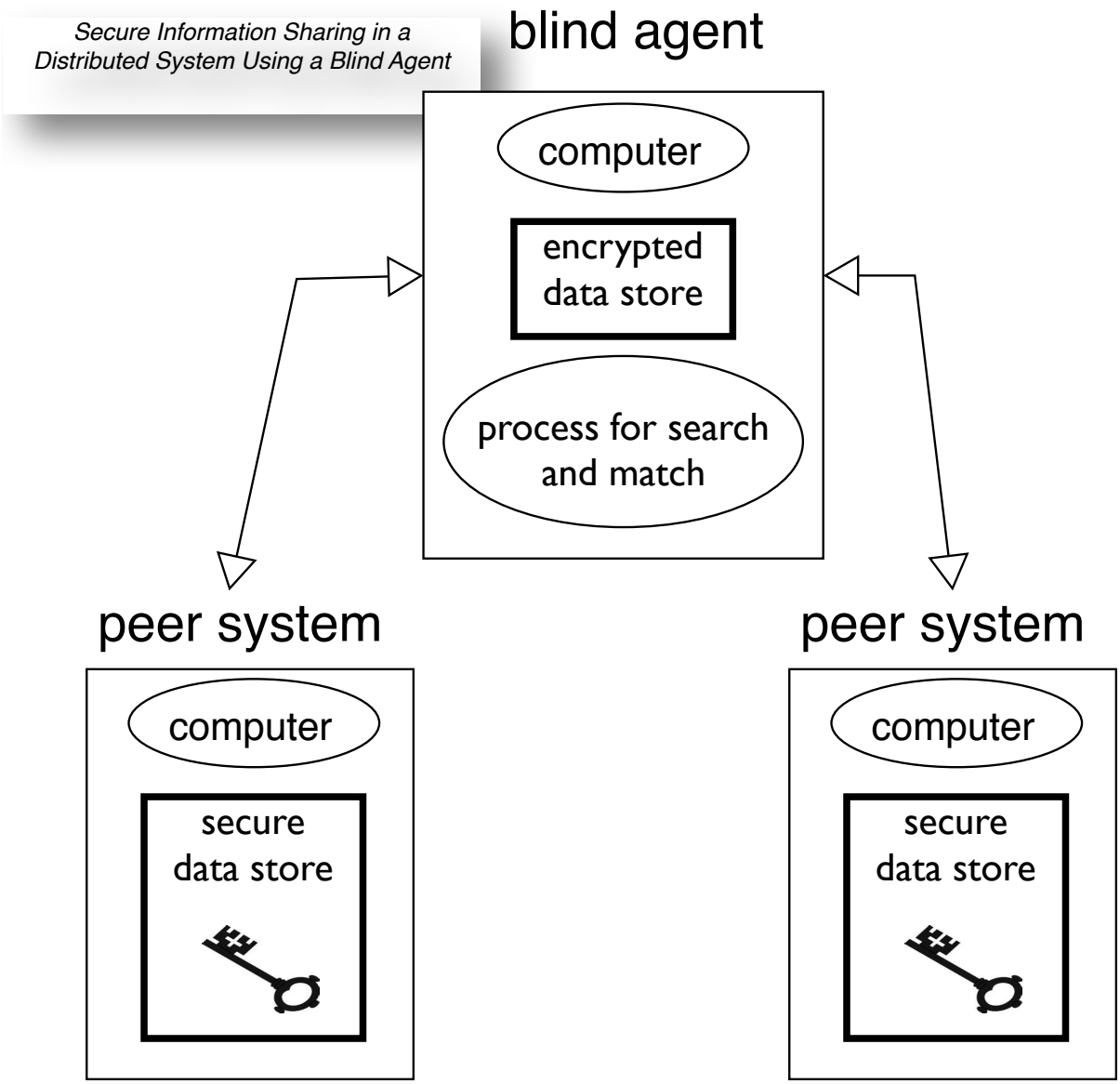


The schematic diagram above illustrates how two or more systems combine with a network to create a distributed system. Right away we see that the distributed information system has multiple management control areas - a potential issue - and an information sharing problem. How can a user on one system take advantage of information stored on another problem?

Also, adding a network creates a new class of user: a network attached user who may access one or the other system or both depending on the user's access rights. In today's enterprise, distributed systems, what information sharing may occur is accomplished through the auspices of network attached users who have access rights into two or more systems. This popular sharing method creates nightmares for the managers responsible for each system. How can they protect themselves from the people and policies in place at other locations in the distributed systems? The architecture laid out in the following will show how to defend against the security risk in distributed systems.

Distributed System with Blind Agent

The following diagram gives the general notion for secure information sharing while omitting many architectural details that will need to be filled in. However, it does illustrate the essential idea that information sharing is based on searching and matching encrypted data in a separate system: the blind agent.



The agent is called a *blind agent* because it cannot decrypt the data; therefore it is blind to what is actually in the data it processes. The architecture demands a separate distributed system for the blind agent to prevent any possibility that rogue software introduced to the blind agent may somehow obtain the encryption key and thus gain access to the data in the clear.

Data security is based on the quality of the encryption algorithm - typically AES 256 but stronger encryption can be substituted. Data security is based equally on the strict separation of encryption keys from the encrypted data which is being searched. Defeating this type of system requires simultaneous compromise of one of the peer systems and also the blind agent server.

Given a distributed system with a blind agent, the peer-level systems are able to use the blind agent to locate useful pieces of information that one peer holds but another peer needs. The blind agent is truly blind and doesn't know what the information is, but it knows it is important. Based on that importance, the peers can share the information through a secure channel. Thus the cycle is the discovery of an opportunity to share, followed by a decision to actually share and completed by a secure communication of a small amount of relevant, high-priority information. Throughout the cycle, the bulk of the sensitive data is fully protected from unauthorized access.

The Distributed System Operates Peer-to-Peer with Special Roles

The diagram above may give the misleading impression that the blind agent should be regarded as a server in a client-server role with the other systems as clients. However, a server in a client-server architecture must wait for a client to call for service. Although the blind agent does provide a matching service, it does more than wait. In the secure information sharing system, there is a workflow that includes blind agent calls to the other systems and calls directly between the systems. Therefore, the architecture is properly characterized as a peer-to-peer distributed system in which one of the systems, the blind agent, plays a different role than the others.

Later, we add a nuance to this story by pointing out it is possible to employ *Message Agent Middleware*, which enables the implementation of a peer-to-peer architecture with software components that insist on being clients invoking a server.

Peer to Peer Communication Uses a Consistent Mechanism

The communication links between systems rely on a consistent, uniform mechanism. Specifically, the systems exchange messages with a document attachment. This mechanism ensures that the distributed system has an ability to handle many simultaneous data sharing operations without delay or complexity. To achieve this goal, the message portion contains attributes that fully specify the particular sharing operation to which the message relates. The document attached to a message may

contain sensitive information so the document is always strongly encrypted as we discuss below in the section entitled “The Encryption Layer Design Motif”. Finally, it is important to note that the encrypted document is also digitally signed by the sender. The message attributes contain the name of the sender as well. The first step in processing every message is a comparison of the apparent sender with the digital signature. If the two are not consistent, the message is rejected.

In summary, the communication mechanism has the following properties:

- One peer sends a message to another
- The messages contain administrative attributes including the name of the sender
- The message has an attached document which is always encrypted
- The encrypted document is digitally signed by the sender

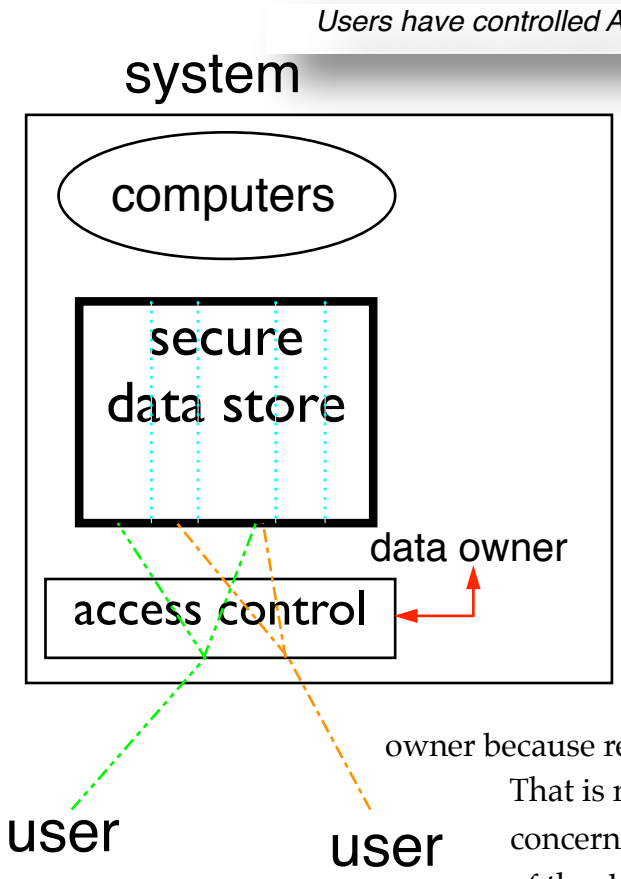
Please note that it lacks the following properties which are considered dangerous in a secure system:

- Message passing is the only mechanism. There is no shared data. No peer is required to offer data services to another peer in the system.
- Each message moves one way. The sender does not wait for a reply; therefore, an availability problem with the receiver cannot propagate back to the sender.

System Users versus Data Owners

The preceding discussion considers only users and their systems - which is an oversimplification. In a modern secure system, no user has full access to all the

sensitive data not even the system administrator. Instead the system is provisioned with access control mechanisms that limit a user's access to data. The access control mechanisms will, in effect, partition the sensitive data based on certain attributes. The access control mechanism will evaluate each user's right to see data by comparing the user's access privileges with the data's access attributes. The access control mechanism is administered by someone or some group. We refer to the access administration as the "data owner". In many organizations, it may be hard to establish the data owner because responsibilities can be unclear or diffuse. That is not a good thing. In fact, security concerns should dictate the careful assignment of the data owner role.



The notion of data owner is crucial to understanding the secure responsible system for information sharing in a distributed environment. In a distributed environment, there are many independent data owners. The sharing function requires cooperation from all of the data owners.

System Failure

The new architecture described in this document enables a new and unprecedented capability for safe, responsible information sharing. It is too much to hope, however, that it can plug glaring deficiencies in current systems. These deficiencies lead to system failure and the new architecture does not prevent failure caused by well-known, tried and true mistakes. Consequently, it is important to point out common system failure modes to foster a better understanding of the distributed, secure information system. There are two major failure paths with alternate failure mechanisms on each failure path. We consider each in it turn.

Failure to Protect

If a chain has a weak line, adding stronger links somewhere else does nothing to strengthen the chain. Consequently, strong physical and network security is absolutely essential. There are many aspects to security but two are especially relevant to this discussion.

First, all systems are vulnerable to access privilege escalation. When this occurs, users are granted broader and broader access rights in the hope - a quite reasonable hope actually - that they may find something to help the mission. In the past, access has been made broader as a few success stories appear (think for example of the expansion of SIPRNet) but then access is perceived as a vice after a failure to protect (think for example of the Bradley Manning affair). With regard to this failure mode, we can say that the architecture does not prevent failure, but it makes it much less attractive to flirt with the idea of failing to protect private data in the hope of bettering the chances for information sharing. With the new architecture, we can achieve valuable information sharing without the risk of broadening access rights.

Second, all systems are vulnerable to malware introduced into the system. Malware can enter over the network or it can be carried into a facility on the extremely popular removable media. Of the many recommendations one may make in regard to this failure mode, we want to point to white-listing. White-listing is the use of anti-malware software that detects all software in a machine and compares it to a list of authorized software. If a software unit is found and it is not listed on the approved list, then it is deleted. Naturally, the actual digital signature of each unit must be checked to prevent substitution of a forgery. White-listing is mentioned here because we will refer to the idea again in the discussion of one of the design motifs: zoned defense.

Failure to Share

Today, it is rare when agencies routinely achieve high-impact information sharing. They fail to share because of stove-piped systems and incompatible data formats. These are serious issues that must be addressed.

The connectivity issues of stove-piped systems will likely be addressed only through new connections between systems. If one plumbing system used copper piping and another used iron pipes, we might reasonably expect to join them with an adapter. In that spirit, the sharing architecture described here can act as an new adapter to provide that new connectivity.

A more serious problem is that the stove-piped systems use different, immiscible information formats. A solution to data interoperability issues has been delayed too long by over-promising on the part of vendors. While standards exist, like XML, the standards are only for syntax rather than semantics. Software vendors are not positioned to solve the semantics problem with mechanical manipulation of data. Overcoming the compatibility barriers is the job of data owners. They must successfully negotiate a mutually-comprehensible semantic-interpretation of stored information.

The sharing architecture described here resolves the security protection issues around sharing. It does not resolve the data incompatibility issues. We expect some progress on that problem through the standardization efforts under the NIEM banner (National Information Exchange Model). Much will also depend on a few heroes who actually put standards in place over the opposition of the status quo.

The Session Group Viewpoint

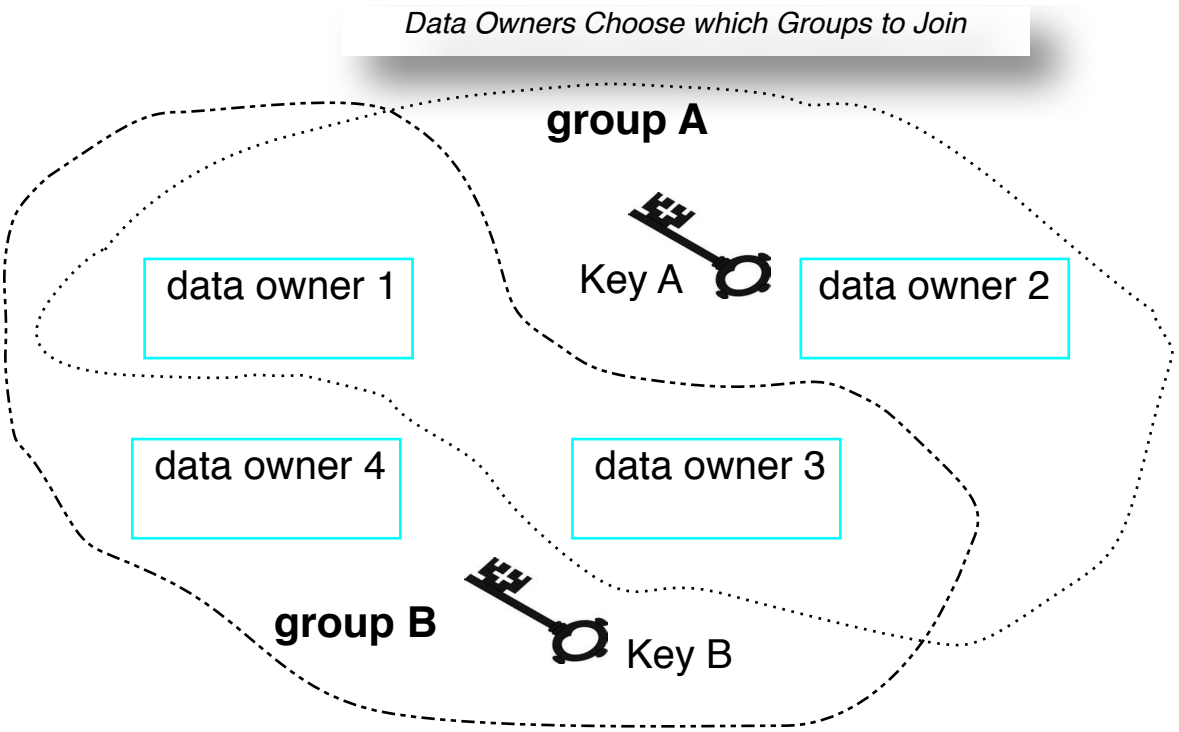
As we saw in the foregoing discussion, the security of our information sharing architecture requires the strict physical and network separation of encryption keys from the data that is concealed by the key. If the two ever meet again, the data is easily decrypted and read. A sound architecture will provide for convenient but foolproof separation of key and encrypted data. The conceptual architecture provides the organizing concept of the "session" to serve this purpose.

There are many sessions; as many as needed. Each session has a finite duration. During the session there is an associated encryption key called the "session key". Information is encrypted with this key and made available to the blind agent in encrypted form. The blind agent can then compare, search and match encrypted information from several data owners. At the end of the session, the encrypted information is destroyed, with the possible exception of a secondary information product: the encrypted audit trail - a topic that can be deferred at this point in the narrative.

Session Membership

The parties that contributed encrypted information to a session are referred to as the "session group". Ideally, every party has the opportunity to review the identities of other parties in the group and opt out of the group without submitting information. We say ideally because that is a rule or convention but it is not enforced by the architecture in the same way that information security is enforced. However, because group membership is voluntary, many session groups may be required to accommodate the preferences of the parties, that is of the data owners.

For example, the following figure illustrates that two groups (A and B) are needed when two of four data owners (2 and 4) don't want to cooperate. Group A includes owners 1, 2 and 3 while Group B accommodates 1, 3 and 4.



The figure above illustrates that each group has a separate encryption key that data owners in the session group use during the session. As explained above, the session key changes between sessions; moreover, a different key is used for each group. The start and stop times are also set independently for each group. It follows that any data owner who participates in two groups will need to perform two encryption steps, one for each group. This feature of groups can be time consuming but common sense indicates a natural limit to the number of groups. If there is really a lot of mistrust among the parties, the mistrust might result in a proliferation of session groups. More likely, however, mistrust will result in the complete breakdown of cooperation. In most aspects of life, cooperators start out in small groups, learn who they can trust, and gradually add trustworthy parties to one group. For the in between cases, multiple groups coupled with the power of cloud computing should handle the load. However, the mention of cloud computing is too early here; we consider the topic later in the story.

Session Roles

All the members of a session group are equals or peers; however, there are two roles that must be assumed by one peer acting in a role. They are the following:

- **Session Key Originator** - This role is assigned to one of the members of a session group. When the session starts, the originator creates a new, unique session key and distributes it securely to other members of the session.
  (Note: secure distribution of the keys is explained later in the *Encryption Layer Design Motif* section)

- **Session Membership Coordinator** - This role is played by one of the peers who essentially recruits the rest of the group. If the role is played by one of the data owners, then the blind agent must approve the choice of session membership coordinator. Alternatively, the blind agent could play the role of session membership coordinator and recruit the session group itself.

We expect that commercial applications will lean towards assigning the coordinator role to the blind agent because there is more of an assumed client-server relationship in commercial settings. Government agencies, on the other hand, will probably lean towards forming their own groups and electing a coordinator. In principle, the session membership coordinator might be an agency separate from both the data owners and the blind agent.

The State Transition Viewpoint

In an architecture that is driven by messages, it can be useful to view the temporal dimension of the architecture by means of event diagrams. In this system, the arrival of a message is an event and the recipient will respond to that event. The actual response is determined as follows.

The distributed system for secure information sharing includes software components that implement state machines. These state machines are configured to respond when prompted by a message according to the detailed system design. The details of this response include aspects derived from an organization's policy; therefore, the state machines are configurable as part of the installation at a site.

It is helpful to consider a small aspect of the state machine configuration to illustrate this important part of the detailed design. In the following we consider a simplified scenario for the distributed system.

Figure 1 on the following page shows an event diagram for part of the process in a simple system with just two data owners who wish to share data securely. An event diagram uses vertical lines to illustrate the timeline for each independent actor in the scenario. The scenario illustrated in Figure 1 adopts the option that the blind agent will be responsible for the enrollment of members in the session group. At the start of the interaction shown in Figure 1, Data Owner B is enrolled already. At the point in time (1), Data Owner A sends a message to the Blind Agent requesting membership in the session. The Blind Agent accepts A and also notifies the other party B. Now the two parties, A and B must approve each other before the session group is complete at point (2) in time.

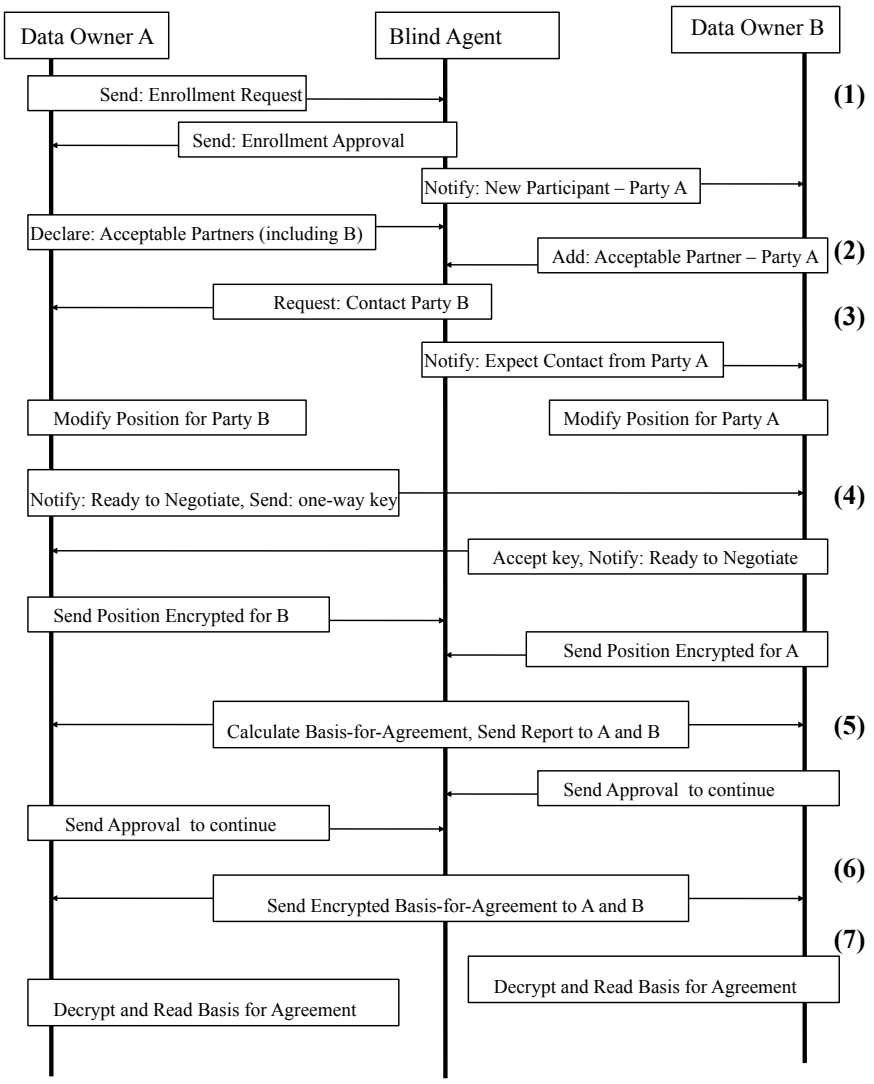


Figure 1: Transition Diagram for the Method of Operation. The diagram shows the enrollment of Data Owner A in the session and the subsequent negotiation between A and B. The enrollment of Data Owner B and the initial preparation of negotiating positions are not shown to simplify the diagram.

With the session group complete, the blind agent starts the scenario by selecting one of the parties to be the Session Key Originator. In the example illustrated in Figure 1, the blind agent picks A as the Originator and sends a message to A and B at time (3). Both parties get ready for the matching session by preparing their offering for the encrypted matching step. In addition, Data Owner A prepares the session key and sends it to Data Owner B in a secure message at time (4). Next, the two data owners encrypt their selected information with the session key and send it to the blind agent.

In the next step, the blind agent will use matching, searching and other algorithms to calculate what we like to call the "basis-for-agreement". For a moment, let us consider why such language is appropriate. Data Owners have independent operations, independent security and separate management. While their missions may overlap, they will certainly see differences between themselves in regard to goals, constraints, and policy. The differences between data owners are the reason it is hard to develop a culture of information sharing. While it is common to attempt to force a sharing culture into existence, it may be more successful to treat sharing as a negotiated agreement between two adults. The blind agent is assisting with negotiation of a sharing agreement. It does this by finding information items belonging to multiple parties that relate to the concerns of both parties. Thus, it is in the interest of both to share. These matching items are referred to a "basis for agreement" because they represent a small subset of the full information holdings that while small is directly and immediately important. Thus, the parties can agree to share just important items without obligating themselves to giving away all their data without any restriction.

The calculation of a basis for agreement leads to notification messages sent from the blind agent to both data owners at time (5). The event diagram of Figure 1 then illustrates another optional choice for the process. Both parties need to give approval before any sharing occurs. When both announce their approval, one of two steps may occur next. First, the two parties simply send each other the data that each needs. This is direct and keeps the blind agent out of the exchange. However, there is a second option. The blind agent can take note of the approval and send the relevant data items - fully encrypted as the blind agent has them - to both parties. That is what happens at time (6) in Figure 1. Unlike the blind agent, the two data owners both have the session key; therefore, both data owners can decrypt the shared data at time (7).

Many variations on this scenario are supported within the architecture and indeed Figure 1 is far simpler than a typical interaction. However, the figure should illustrate the operation of the system in the temporal dimension. In a realistic system, the operation of the system in the temporal dimension is specified by configuring the state machine components of the system.

The Analyst's Transaction Viewpoint

The architecture description so far emphasizes systems but the whole point of the system is assist people to do their job and fulfill a mission. The team usually has people who are especially skilled at dealing with information. Let us call them "analysts" for this discussion. The key point is that the work of analysts will not change with the introduction of a secure information sharing environment; it will simply become more productive and successful.

The truth of the matter, however, is that the analysts conduct different kinds of transactions on the information system. Some kinds of transactions will benefit more from the improved environment than others. For this reason, we will establish a crude taxonomy of the transactions and discuss the effectiveness of blind encrypted data matching in the context of these transaction types.

Here is a list of five transaction types. We discuss each below but the list summarizes how well encrypted data matching can support the transaction type.

X Instant, Interactive Transactions - unsupported

✓ Delayed, Interactive Transactions - supported

✓ Alert Driven Transactions - supported

✓ Case Centered Transactions - supported

? Statistics Based Transactions - no general conclusion

Instant, Interactive Transactions: occur when the analyst poses queries at a work station and receives an immediate response from a search engine that has access to the sensitive information. Because the query/response cycle is fast, the analyst can pose a sequence of queries perhaps staying with one topic. Consequently, an analyst's productivity depends on having a low system latency - that is a fast response. Already this can be problematic when the stored information is

voluminous and even more problematic when it is distributed. When blind data matching is introduced in the system to provide security for sensitive data, the latency in receiving a response will make instant, interactive transactions too tedious.

The success of information transactions in an information sharing environment is doubtful in any case because responsible information sources will be reluctant to provide interactive browsing into their sensitive data. Allowing such browsing is very risky.

Delayed, Interactive Transactions: are sessions in which an analyst starts a query and then sets it aside while the system finds an answer. For this type of transaction, an analyst's productivity hinges on launching many parallel, simultaneous transactions perhaps on completely different topics. A well-designed analysts-workbench will manage the parallel transactions and present results when they are available. This form of transaction is suited to the new environment because the latency of the matching sessions is compensated by the parallelism in the transaction architecture.

With suitable controls and reviews, responsible data owners may cooperate to allow this type of transaction. It differs from the instant transactions because sharing can be controlled by monitoring the potential transaction before it is completed. A valid query will contain highly specific attributes that allow a data owner to decide whether the analyst is pursuing a valid line of investigation or fishing randomly for private data. In addition, the data owners can block a transaction if the query is overly broad and requests too much information. While controls like this represent matters of policy and while such controls must be refined by experience and practice, it is important to note here that the architecture provides choke-points in the flow where control can be asserted. Because the blind encrypted data management system preserves the autonomy and independence of the data owners, it is likely that they will cooperate with the sharing operation.

Alert Driven Transactions: A good information analysis system is proactive and continually monitors the growing data archive to find new pieces of information that should be considered by an analyst. The analysis system will therefore include algorithms and agent processes that search for events and circumstances of interest. When results are found these appear as alerts on an analyst's workbench. In practice, an alert driven transaction is similar to the delayed interactive transaction

in that the alert corresponds to the delayed response to a query. The practical difference lies in the ownership of the alert agents. The software agents that find the alerts are the collective responsibility of all the software analysts. When a software agent finds something noteworthy it decides which analysts should receive the alert. This type of transaction is well suited to the blind agent architecture.

Case Centered Transactions: are long duration transactions during which a portfolio of information is gradually built from many other transactions. Frequently, the results found early and added to the portfolio will guide subsequent queries. Also the developing case may display patterns that can be used to extend the search for related information.

A good analysis workbench should support case centered research. The blind agent architecture supports the case centered approach to the extent it supports the underlying short transactions. In general, this type of transaction is well suited to the blind agent architecture.

The case centered transaction raises an important point: research on sensitive private data is itself a private, sensitive matter. In the course of the investigation, an analyst guiding the case-centered research will accumulate sensitive data as well as hypotheses that may or may not be supported by evidence found later in the investigation. Revealing the content of a case prior to conclusion would be unhelpful and possibly illegal. On the other hand, suppose two analysts working in different organizations are working on two cases that pertain to exactly the same events or interests. Should not the two analysts know about the similarities in their cases? The blind encrypted data matching method is ideal for safely finding similar cases without revealing the contents of either. However, once the analysts are notified about the similarity, they may wish to share, cooperate and join forces.

Statistics Based Transactions: There are workstations for analysts that are provisioned with software agents and algorithms that study the available information for statistical anomalies. These workstations report any anomalies to an analyst in both written and graphical form. For example, textual analysis might reveal a small subgroup of apparently unrelated people who say the word “ni”. As this is statistically unlikely, an analyst might be alerted to study the situation because there might be a subversive organization of “Knights who say Ni”.

Given the risk from as yet unknown threats, the search for statistical anomalies has to be considered part of routine analysis. At a transaction level, the statistics based transaction has elements of the alert driven transaction and the case centered transaction. However, it is not clear whether blind encrypted data matching systems will support the statistical transactions. The uncertainty arises because the number of known encrypted data matching operations is limited; therefore, the statistical analysis of data is constrained. This type of transaction is an area for future research and may require one or more of the following technical approaches: homomorphic encryption, an extension to the architecture to support captive agents, or reformulation of the statistical algorithms to use only the available encrypted data matching operations.

In summary: analysts will continue to use workbenches without disruption but their scope of operation over distributed information will improve after the introduction of secure, responsible information sharing through blind encrypted data matching.

The Layered-Encryption Design-Motif

The distributed system is driven entirely by the arrival of messages. Naturally, there is a scheduler or perhaps a human operator somewhere who kicks off all the activity by sending the first message but apart from this startup step or a potential system-wide shutdown command, the activity is automatic and message driven.

When a message arrives, it will usually have an encrypted document attached to the message. An attached document is protected in transit by layered encryption. By layered, we mean that there are multiple encryption keys that secure the document and multiple decryption steps are needed to fully reveal the content. In an installation with a zoned defense, the multiple steps will even be conducted on separate machines separated by firewalls, thereby increasing threat resistance. In this section we explain the encryption layers. The layered encryption design is one of the significant design motifs of the architecture.

For the blind encrypted data matching method, the characteristic part of the design motif is the innermost encryption layer - the layer protected by the session key. In this innermost layer, the document is encrypted by a partial encryption procedure. To understand partial encryption consider that every document is composed of shorter sequences, i.e. either statements or sentences. Every statement or sentence has a recognizable grammar rule that defines its structure. A computer can recognize the grammar rule that applies to a sentence by looking at certain structuring words.

For example, it distinguishes nouns and verbs and pays special attention to the placement of conjunctions and prepositions. In a partial encryption, the conjunctions, prepositions and punctuation are not encrypted. Often a single verb is left unencrypted to convey the context of the statement so that it can be compared with statements of a similar or related context. On the other hand, nouns are always encrypted - especially names, places, dates and others - as well as adjectives and quantifiers. Special consideration is given to numerical quantifiers through the use of order preserving encryption algorithms. The outcome of partial encryption is that documents still contain statements that can be distinguished from each other and the structural grammar rule for each statement is apparent from the partially encrypted material. The outcome resembles a redacted document as we illustrate with the following example:

```
offer:
   accept: DIS
   for: IBM
   cash_per_share: [    0.0000,    77.6159]
   number: [7507, 60060]
   interval: 12
   ref_id: 177379
end
```

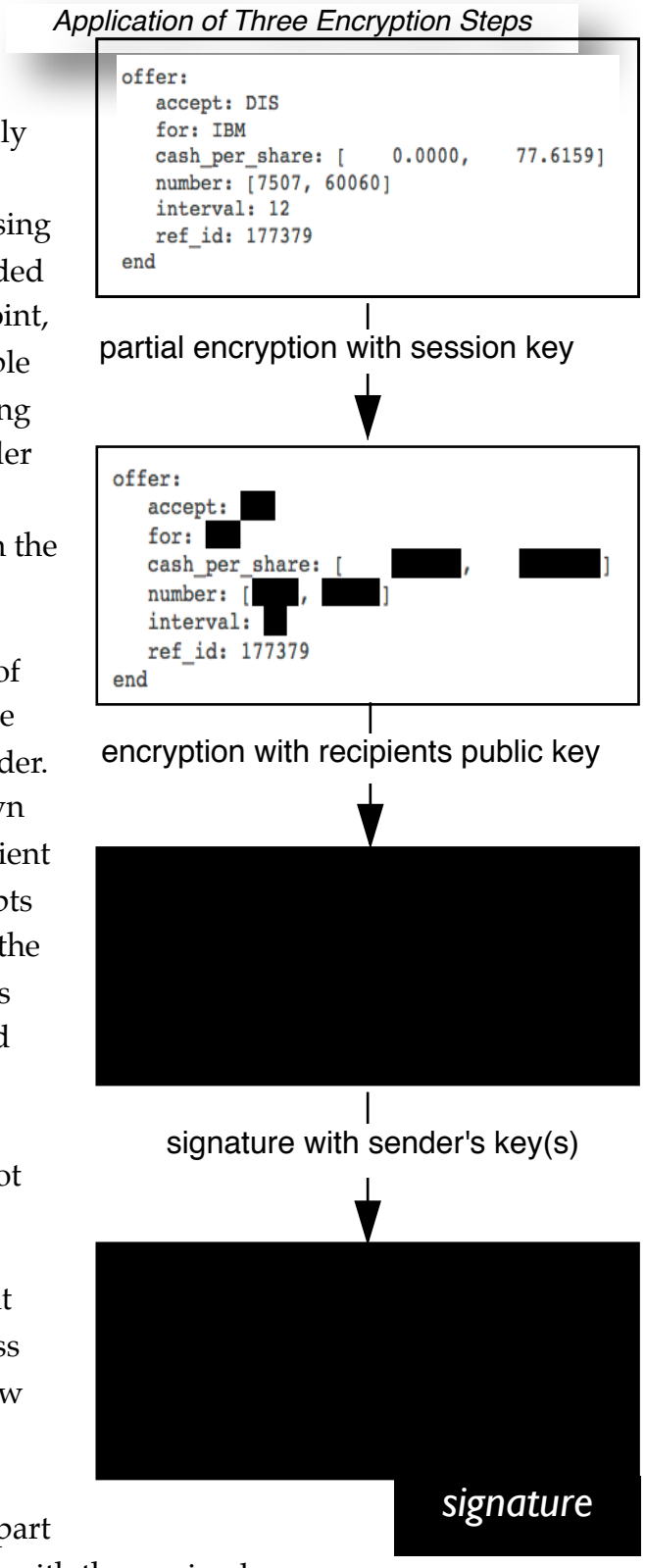
This example is taken from a full discussion of blind encrypted matching that appears on the WWN web site. This statement appears in the stock swap example. It uses a fairly obvious grammar that was invented for the purpose of illustration. In the grammar, the keywords that are followed by a colon mark are left unencrypted while the symbols following the colon are encrypted as illustrated by the partially encrypted statement shown next:

```
offer:
   accept: [redacted]
   for: [redacted]
   cash_per_share: [    [redacted],    [redacted]]
   number: [[redacted], [redacted]]
   interval: [redacted]
   ref_id: [redacted]
end
```

The preceding illustration looks very much like a redacted document but that is misleading to a certain extent. Redaction simply destroys information. An encryption preserves it in a secure way. The redacted field is no larger than the original clear field. An encrypted field, on the other hand, is much larger. The output of encryption is always a large symbol regardless of the size of the original symbol. We mention this technical point because some critics argue against encryption on the basis that it increases the volume of data that must be processed. Thus, if efficiency and low cost are the main concerns and security is not so important, then by all means forget encryption and just publish the data! It is worth noting that the current networks are sized to accommodate vast numbers of users watching videos. Unless an information sharing system tries to capture and encrypt all that video, it is likely that secure encrypted processing is just a blip in the big picture.

We can now explain the layered encryption by walking through the process starting from an unencrypted information document in one secure location in the distributed system and ending with a document that can be attached securely to a web document and sent over the network to another secure location.

The first step uses the session key to partially encrypt the document. This encryption leaves some fields clear so that they can be processed by the blind encrypted data matching algorithms. Next, the document is fully encrypted using asymmetric key encryption (public key encryption) using as the key the public key of the intended recipient for the document. At this point, the document is completely unreadable unless one possesses the corresponding private encryption key. The only holder for that key is the intended recipient. Thus, it is impossible to eavesdrop on the message .



The next step creates a digital digest of the encrypted document and signs the digest with the private key of the sender. This encrypted, digital digest is known as a digital signature. When the recipient gets the message, the recipient decrypts the signature using the public key of the sender. Then the recipient recalculates the digital digest of the still encrypted document. If the two match, then the recipient is certain that the sender produced the document and it was not modified in transit.

Readers who are familiar with current practice will recognize that the process steps outlined above follow well-know standard practice for secure communication using public key encryption methods. The only novel part is the first step: the partial encryption with the session key.

It is also worth noting that there may be two digital signature on the document. If the blind agent does not organize the session groups then the blind agent may not be sure that the sender is a member of a session group. If that is the scenario, then the authorized session membership coordinator will create a unique public key pair for the session and session group. Members of the group receive the private part of the key pair while the blind agent receives the public key part. With this key distribution, the document is signed with the private part of the session group key and the blind agent authenticates the document using the public part.

The layered encryption is admittedly complex so it is helpful to identify the broad features as they concern the distributed architecture. First, keep in mind that the whole purpose of the public key system is the positive identification of both the document originator and the document recipient. The keys involved are effectively identity keys. On the other hand, the purpose of the session key is to totally obscure the information so that neither the blind agent nor any party that intercepts the document will be able to compromise security and read the information.

With that in mind, we can say that two variations of layered encryption were described in the preceding:

1. The 2-ID key + 1 Session Key system. For this system, the blind agent organizes and recognizes session groups.

2. The 3-ID key + 1 Session Key system. For this system, the documents are signed with the session group ID key as well as the sender's ID key in order to authenticate the session membership of the sender.

The use of cloud computing introduces two other variations which will be discussed later:

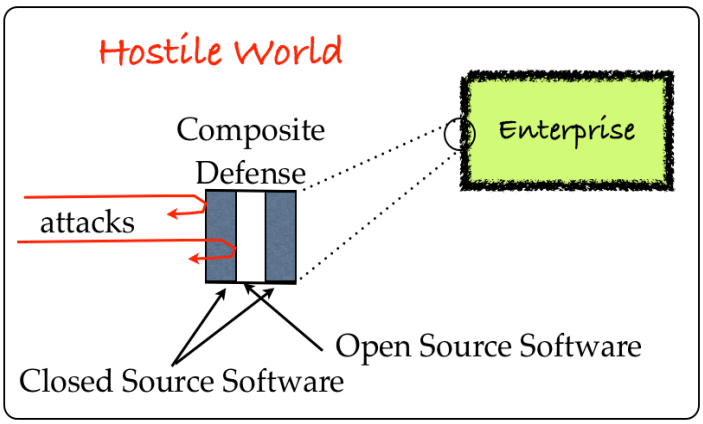
3. The 2-ID key + 2 Session Key system.

4. The 3-ID key + 2 Session Key system.

The Black/White Composite-Defense Design-Motif

Although no modern warfighter would adopt the cast iron breast plate of a conquistador, modern cyber-warriors often trust in "more cast iron" to protect against attack. We disagree. The best cyber-defense is a composite of security software with various properties which complement each other to resist attacks that one material alone could not withstand - much like the composite amour of tanks and soldiers.

Combine Open and Closed Source Software for a Black/White Security Armour



Previously, we described how the security software depends on the careful separation of secured data and encryption keys. That aspect of the software system is actually provided as open source software. The advantage is that the security strengths are obvious in the open specification of the software. Moreover, the code is available for inspection by academic investigators and the public so that the best minds can find any flaws in the mechanism before it is trusted in the field. We call this "white software". In contrast, most security software is closed, proprietary and secret. Part of its strength relies on the idea that the enemy can't know or guess how the closed or "black software" operates. On the other hand, a clever enemy can sometimes defeat closed software and successfully perpetrate an attack in spite of the barriers thrown up against white-hat code reviewers.

Most organizations are familiar with security breeches that happen because of flaws in proprietary, commercial PC operating-system software. In the open source software community, there are fewer surprises because many friendly eyes have checked the software for flaws. Best of all is a combination of open and closed source software - a black/white software composite. This composite software protection takes advantage of both closed and open software and maximizes the protection from outside attack.

The Zoned-Defense Design-Motif

One advantage of the clean, message-passing distributed-system architecture is that there are clear and simple data flows that can be defended by a zoned defense if desired. A zoned defense is a series of nested security zones separated by network firewalls. An intruder must negotiate not one but several firewalls and subvert not one but several machines. We can further complicate the intruders problem by zoning the software as well through these steps:

1. Divide the software function into a sequence of steps

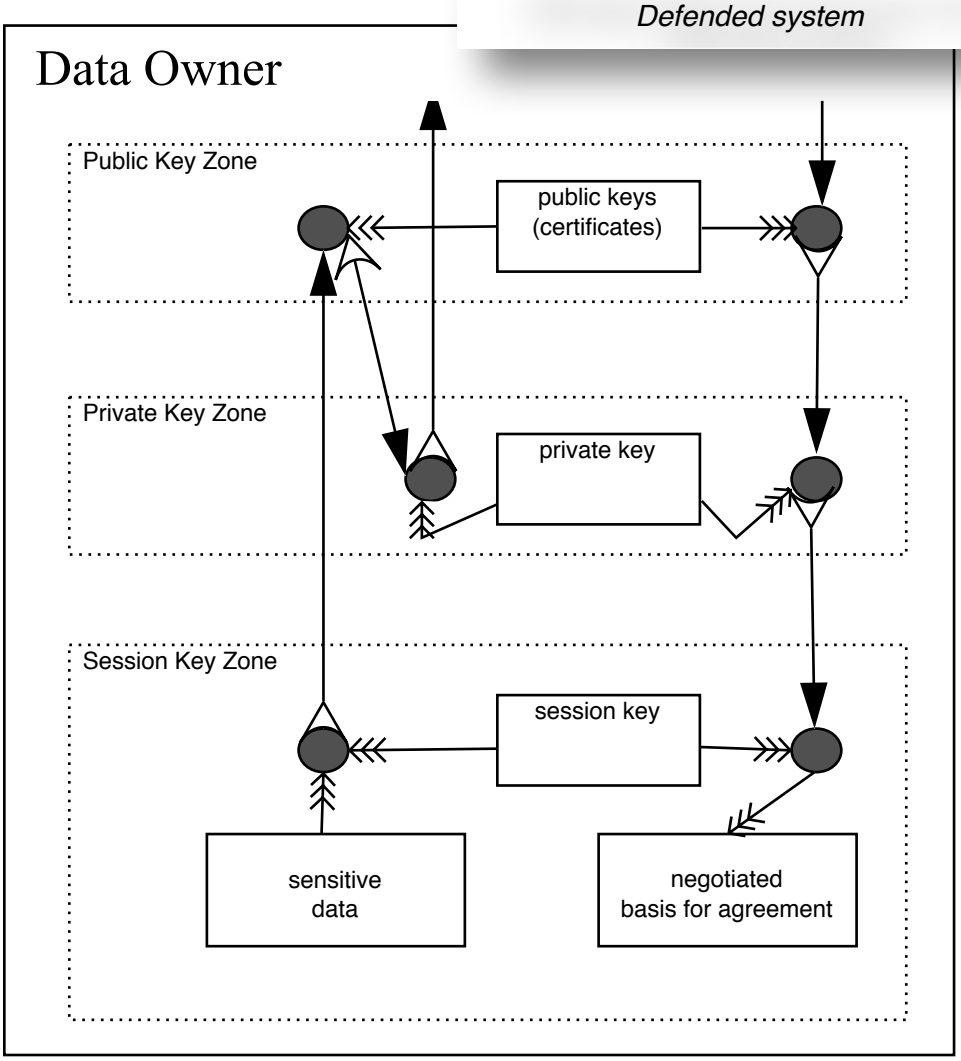
2. Allocate the steps to different physical machines. Separate the machines by firewalls that regulate communication to the approved channels for the software function.

3. Create a separate software distribution for each step and include in each distribution only the software necessary for the step - do not include the complete software suite. Install the distributions on the different physical machines. With the physical separation of the software, malware entering the machines will not have access to the functions needed to build unauthorized paths into or out of the system.

4. Add white-list anti-malware software that operates by listing all the valid software modules in each zone, detecting what modules exist in the zone and then eliminating any unauthorized or modified modules.

There are many ways to zone the software and the following figure illustrates a three zone separation that is based on separating encryption key information so that no single zone has all the encryption keys.

Message Paths into and out of a Zone
Defended system



The Extension for Message Agent Middleware

In a large, peer-to-peer distributed system there is an ever present possibility that one or more of the systems may go off line or the network may be broken temporarily. Messages are moving throughout the system but will fail to reach their recipients if the recipient or network is off line or running slowly. In a system like this, problems at one location may propagate throughout the system by impeding the message traffic. In baseline system architecture, this availability issue is handled at an implementation level as follows.

In each system, multithreading is used so that a communication delay will block one or two threads but other threads continue their processing. Moreover, the state machine objects are implemented with object persistence and the state is backed up continually to a database so that a local machine failure will require no more than a restart followed by a restoration of state from the data base.

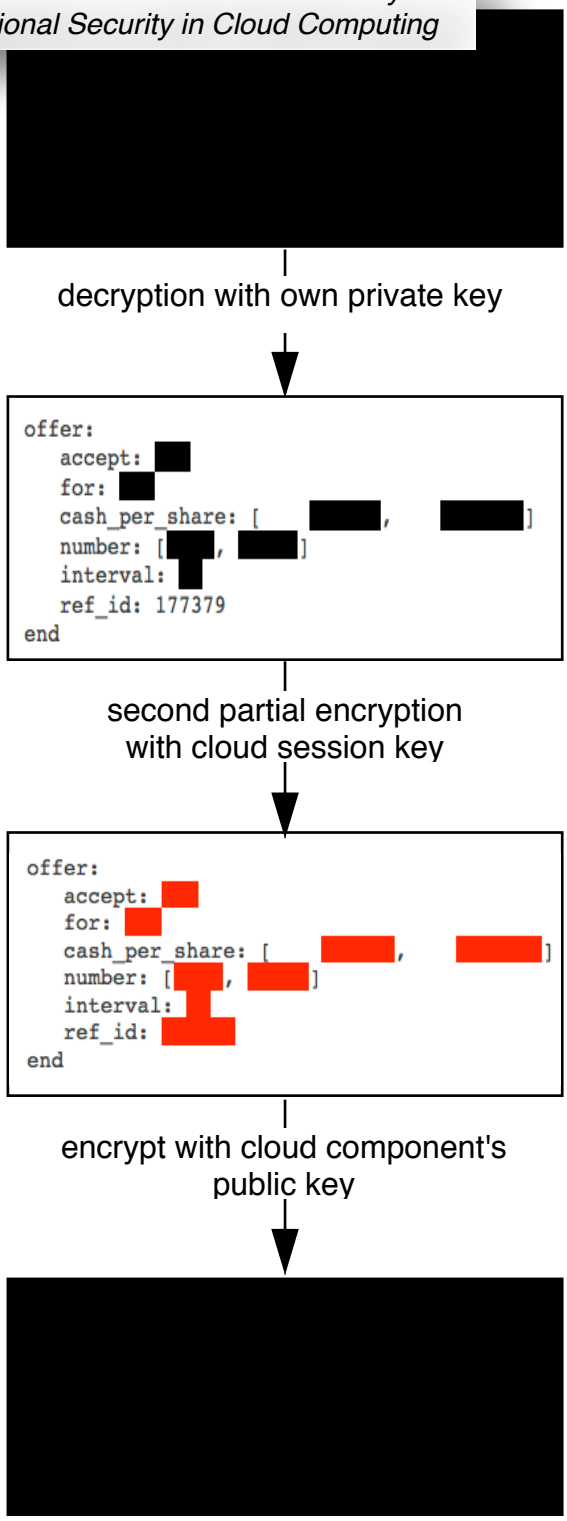
This type of implementation is quite feasible using standard Java EE components. Before such components were available from reliable commercial sources, it was standard practice to handle the availability problem with a middleware solution. A middleware solution is an architecture where the one-to-one peer communication patterns are broken by an intermediate software component known generically as the Message Agent Middleware. The function of this component is logically identical to a mail service. Just as we expect an e-mail to be delivered in spite of any transitory problems, we can expect a message to be delivered by the middleware as soon as possible.

There are arguments for and against a middleware implementation. The most important consideration may very well be whether the enterprise is already using a messaging solution that it trusts. If so, the path is obvious. Otherwise it is not obvious that it is easier to build a reliable middleware solution than it is to build a reliable peer-to-peer system on the basis of Java EE components such as Java RMI in multithreaded applications.

The Extension for Cloud Computing

The opportunities for information sharing are greatest in very large organizations with many employees and large data stores. But large applications will require considerable computing power. Cloud computing is a flexible and popular method for providing additional computing resources. A large number of computers are located in the cloud and assigned dynamically to meet the needs of demanding applications. On the other hand, each computer in the cloud also represents a target for intruders who want to gain access to sensitive data. One should not simply spread sensitive data over many computers without additional security. This security can be provided by a second session key.

Application of a Second Session Key for Additional Security in Cloud Computing



The process is illustrated to the right. The blind agent receives encrypted documents intended for encrypted data matching. The blind agent removes the layer of encryption that requires the agent's private key leaving just the partial encryption with the session key. At this point, the blind agent decides to use the cloud for the actual match processing. However, before it sends the data out, it adds an additional partial encryption step using a special session key that it creates for itself. Then the doubly partially encrypted information can be sent to a computer in the cloud. Of course, the blind agent will secure the document in transit by encrypting the information

again with the public key of the intended recipient - a computer in the cloud.

At the end of this additional encryption, we have information that is encrypted with a session key that is known only to members of the session group and a second key known only to the blind agent. Breaking this system will require penetrating not just a cloud computer but two other computers where the session keys are stored. This presents a formidable obstacle to intruders.

The Extension for Adjudication of the Audit Trail

The architecture provides an independent agent, the blind agent, to mediate safe information sharing between other independent agents, the data owners. It might appear that the blind agent is ideally situated to maintain a record of all sharing operations and furthermore that the agent could assist with investigations of those same operations and even adjudicate any disputes that arise out of them. Unfortunately, the blind agent can do no more than gather statistics on how much information is shared between which parties. All other information is encrypted and unusable for the blind agent. The BEDM data protection is so strong it complicates dispute resolution.

To give an example of a dispute, suppose one of the data owner participates in a negotiated agreement to exchange concealed data. As a result, that owner receives matching information from another data owner. Suppose further that one data owner then misuses the data acquired from the other data owner. If data misuse occurs, an injured party has no sure means of redress. At best, the injured party can avoid future problems by refusing future cooperation with the abuser. The blind agent holds a record of the negotiated exchange, but that record is encrypted. It could not be entered in evidence in arbitration or in court. The accuser can decrypt the agreement but a judge or adjudicator cannot verify its authenticity.

When disputes arise, they call the integrity of the sharing operation into question. It is unlikely that the data owners will continue to participate in the information sharing enterprise if they have reason to feel injured by the information sharing. Cooperation is even less likely if there is no way for a data owner to obtain justice after and injury.

Most information sharing environments fail because the data owners have reasonable grounds to fear what will happen if they allow outside investigators to tap into their sensitive data stores. If the future secure information sharing

environment has the same problem, the outcome will be the same. Consequently, the information sharing architecture should be extended by the addition of a new peer agent - the adjudicator. The adjudicator is an independent system that does not participate in normal sharing operations. In normal operation, it has no access either to sensitive data or to encryption keys. However, when a dispute arises, one of the parties brings the matter to the adjudicator and the matter is settled fairly and accurately based on the blind agent's audit trail of encrypted transaction records.

Introducing an adjudicator system extends the enterprise architecture by one additional and requires the enhancement of the operational scenario to support dispute resolution involving the blind agent's encrypted audit trail. Additional details can be found in implementation documentation and the following figure, Figure 2. Please note also, that the adjudication system is not available with the standard software distribution at the time of writing.

Figure 2 - Event Diagram for Resolution of a Dispute by Adjudication. One of the data owners as plaintiff asks for adjudication. The Adjudicator requests the relevant portion of the audit trail. The plaintiff provides a copy of the session key which encrypted the data contained in the audit trail. Finally, the adjudicator renders judgement based on the content of the audit trail.