中庸

The Third Way to Data Privacy and Sharing – Concepts, Tools, Applications

Hypatia Burbidge

Hypatia.Burbidge@gmail.com

The Third Way to Data Privacy and Sharing – Concepts, Tools, Applications

Table of Contents

Cover Photograph by Giusepe Milo CC [i]

Forward

This work starts with a premise, that there is a safe way to integrate computer systems while at the same time balancing civil privacy with the public government's needs and while balancing security restrictions with the sharing imperative. Accepting the premise, we can envision a future where people and systems cooperate better. Better cooperation will advance our technological-dependent civilization. A reconciliation of private and public concerns will make citizens more comfortable with a state powerful enough to pursue an aggressive national agenda.

Although the subject lies in the future, this book is not fiction. It is computer science. With the technology described here, we will build a future world better than any you can extrapolate from contemporary practices and standards. It hasn't happened yet. It hasn't been tried yet. Thus, the book runs on a line between fact and fiction, but it requires no new science, only implementation and deployment.

I plan to hand you the basic keys to the "new" computer technology. You can take the term "new" under a suspension of disbelief because the components are old and well known. We mix them together and something new emerges. We expect mixing these software components will show an unprecedented emergent property: a high level of successful, cooperative activity conducted in a broad society of computers surpassing anything achieved to date.

That anyhow is the vision. The book provides concepts, tools and examples. Any additional talk of emergent properties and the evolution of cooperation will be relegated to a supplemental chapter in "Supplemental Topics". What you actually have here is a serious attempt to convey the computer science behind the future society of systems.

About the Author(s)

The name, Hypatia Burbidge, is a nom de plume. The initial author of this work is not trying to hide. You can check him out on LinkedIn[1]. The decision not to publish under my own name is driven by the fact that the book is incomplete and unfinished in the form you see here. But it can be finished. It just takes more than one person. So my dream is that the work is carried forward by a collective of people who share the goal of a collaborative society of systems arising out of today's Internet. An architype for this collective would be the Nicholas Bourbaki group of French mathematicians.

If you are interested either in getting a notification when new drafts of this document are available or you would like to contribute to the work, please email to hypatia.burbidge@gmail.com.

The name chosen recognizes two great astronomers: Hypatia of Alexandria and E. Margaret Burbidge, most recently of UCSD. Astronomy is always unfinished work. When we regard the hard road ahead, we may draw some consolation that an incomplete understanding of a deep field, e.g. Astronomy or information technology, can still be powerful and beautiful.

[1] https://www.linkedin.com/in/pbaker1

1. Setting the Scene

Setting the Scene — Why?

When interests collide, when issues polarize, when ideals are irreconcilable, society enters a time of fission —a time of division into fiercely-determined opposing camps characterized by a hostility that divides people and thwarts progress. That hostility describes the current state of the field of data privacy — people belong to one camp or another. They favor software that improves privacy or undermines it. In between the extremes there is only a wasteland devoid of proponents and suitable technology.

Tragically, our society relies more and more on privacy technology but watches helplessly as powers battle over which of two polar opposites will prevail: on one side, the privileges of the collective entities — government and corporations —to use our data for their vision of our common good or, on the other side, the rights of individual people — citizens and consumers — to own property and to decide for ourselves how our data is used.

Perhaps this description sounds too dramatic? After all, no blood is being spilled over the conflict and even the lawsuits have been relatively low key. For the most part, people takes sides only on specific issues — is Snowden a traitor or a patriot? is Google a friend or a Big Brother? does Apple obey its customers or the FBI? So far, there is little really at stake for the average citizen. Should the citizen care at all about data privacy?

This book is written because the author cares. Here is why. The battleground is quiet now but this conflict must inevitably heat up as improved technology allows the collective entities to intrude more and more on the individual's perceived rights. Also, journalists will continue to discover and publish intrusions on the public's privacy thereby embarrassing both the corporations and governments. Both will seek to preempt action and suppress opposition. You can expect more controversy and more ill-considered legislation. But, most importantly in the author's view, this battleground is a technical no-fly zone — a no man's land. Geeks usually will hack any hard problem but have left this problem entirely unattended. Anyone who is qualified is occupied building better software weapons for one side or the other (perhaps both if the pay is right). Speaking personally however, this geek abhors that vacuum and takes no pay from either side. I want to fill the vacuum and this book is part of that process.

Setting the Scene — What Subjects?

Trust, Loyalty, and Perfection

Living in a society you are connected to others. Your data protection and privacy are not yours alone; they are bound up in social interactions. Trust, loyalty and perfection are three attributes of interaction in a society. Let's examine these terms. In a society, individuals and their computer systems play assigned roles and take actions that keep the society running. Every role brings responsibilities that might be easy to bear were it not for society's complexity. Most actions require help. Every responsible individual relies on others to do part of the work, to provide part of the information, or even to accept part of the responsibility. Thus every individual is part of a complex social web. Every individual in society must <u>trust</u> others to help. Moreover, those who are trusted must be loyal and act as expected. Trust and <u>loyalty</u> are fundamental to social interaction.

Now we must mention that neither people nor their systems are prefect. Both make mistakes. As a result, trust and loyalty are not enough, society expects perfection or a close approximation.

Encryption and the Third Way

Encryption technology is a fact of life today. Knowledge of it cannot be erased. Encryption is widely available, it is simple enough that even small groups can apply it to their operations, and it is strong enough to thwart even the largest code breaking machines. We can take encryption for granted. What we can't take for granted are trust, loyalty, and perfection. All three are suspect and we can be undermined by misplaced trust, disloyal partners, and imperfect execution. This book is about how to apply encryption technology in a world where trust, loyalty and perfection are all suspect.

Given we have encryption; we have the option to encrypt all our data. But that option leaves us completely isolated with no option to shop on line, no on-line accounts with banks and no friends on the Internet. To be a part of today's world — to work, to live, and to socialize on-line —we need to share some data with others. Perfect encryption blocks that. So we compromise and let some people have some data. Once we share our data, we lose control over it. Sure you can write a nondisclosure agreement, pass legislation, or maybe sign an executive order. All that has been done. None of it can repair the damage when arbitrary limitations are simply ignored. In practice then, we are torn between a policy that locks down data leaving it unused and ineffective or sharing data and putting it at risk. Neither policy is a good one. Hence the need for a third way — an option we can select when asked to choose between locking down all data versus allowing the big players to take it at will.

The phrase "third way" commonly refers to finding a middle ground between two political agendas where the advantageous elements of both agendas are incorporated into a consensus. I'm not talking about that. What I have in mind is more the "middle way" — the Guatama Buddha's term for living in the real world being fully aware of and simultaneously striving toward two opposing goals: ascetic union with the divine versus success in this material world. The Buddhist way is not one way or the other way but a third choice. In the same way, the third way described in this book is neither private data nor collective data but a third way. It is a way forward when the other two lead to conflict and deadlock and, yes, suffering.

Setting the Scene — How?

In Part 2, we will introduce the main ideas behind the third way. Briefly, however, we can say that the third way shares specific data or allows its limited use in those cases for which the relevance of the data can be reliably demonstrated according to a plan agreed upon in advance by the responsible parties. The third way operates inside all of the normal security protections provided by the best current practices. Current practice secures data at rest, secures it in transit, and ensures that it is used by a person or agent with a verified "right-to-know." The Third Way adds more security by insisting that the use of the data be governed additionally by a "need to know[2]." Let us compare these two policies.

[2] *Need to know is a well-known and widely applied principle of security; in the past however, it has been enforced via a subjective, manual process. What we do here is an automation of this essential function. For more see Wikipedia: https://en.wikipedia.org/wiki/Need_to_know*

A “right-to-know” policy allows or disallows the use of vast amounts of data because implementation of the policy operates on abstract, general data attributes. The policy asks whether this kind of actor has a right to use this kind of data — kind of actor and kind of data are general classifiers rather than selective specifications. A “need to know” policy is highly specific to the context and asks whether this particular actor at the present moment of time has a right to use a very specific piece of data in the context of the evidence presented with the request.

The difference between the two access policies is dramatic. When “right-to-know” policy is mistaken, it fails big. If “need-to-now” is mistaken, it fails small. In principle, what was just described could be implemented without the third way. But if one examines the real world situation in more detail, then it becomes clear that the contextual information that is provided to the decision-maker — the one who grants or denies “need to know” — is sensitive proprietary information. Thus, a decision-maker introduces another security risk which is unacceptable. In the Third Way, the decision-maker is not trusted with the information. The decision-maker is allowed to see only encrypted information. Both the protected data and the query context are encrypted during the decision regarding “need to know”.

Setting the Scene — Where?

Currently, the Third Way is the road not taken. If we took it, where would it lead? In Part 3 we show three applications that exemplify the destinations along the Third Way. These applications illustrate the concepts of solidarity, vulnerability, and new markets.

Solidarity. Things happen to people that they don’t want to talk about. However, when the same thing happens to many people, they might band together and do something about the causes of their problem. That is solidarity. Fighting against solidarity is the stigma attached to victims and the afflicted. Sometimes silence seems the best way. Our example application discusses rape on campus — an event where victims feel the legal system fails. A related application would be to match patients with medical conditions to treatments or support groups, a situation where a patient might be reluctant to discuss fears and symptoms with a doctor or a group of peers.

Vulnerability to Betrayal. Many difficult situations involve a group of players who while not enemies are distrustful of each other. This situation arises naturally through competition. I want to win. So do you. Maybe we can work together but maybe I’ll be better off taking advantage of you. It is a fluid situation where distrust can impede successful cooperation. Examples abound in the liaison between different intelligence services and during business negotiations.

New Markets. The free-enterprise market system for assets, products, and services is efficient but competitive. But competition and deregulation put some groups at a disadvantage and introduce a substantial systemic risk of market bubbles and flash crashes. Our example application builds a market where the interests of the participants are protected while maintaining competition. This example concerns a fictitious market for stock swaps. The fictitious market for stock swaps emulates the markets for many actual asset classes.

Setting the Scene — The Plan

This section, Part 1, makes the general case for following the Third Way. Part 2 of this book introduces concepts that are necessary to discuss the topic. Part 3 describes potential applications to illustrate how the concepts can be used to solve practical problems. The sample applications discussed in Part 3 may instruct and motivate some readers to solve additional problems in the same way. Ideally, the series of sample applications will lead by induction to a pattern for many future solutions.

Here are a few tips about how to read the book. You've already read this far. Good. If you are primarily reading to decide if the Third Way has something for you, then I'd recommend jumping forward to Part 3 — *Applications*. A look at the applications may give you some ideas. On the other hand, if you want to fully understand the Third Way then I would recommend reading all of the chapters in *Part 2 — Overview of the Concepts*.

The general question is "how to solve the data privacy problem?" In practical terms, the answer is: "write software." Because private data resides on a computer or in the cloud, any solution must be implemented with software.

Software designers and developers will be interested in the design details explained in *Part 4 — Design Considerations*. This specialized audience may also wish to evaluate and adopt implementation components discussed in *Part 5 — Tools*. While none of the active components in the public toolset is production quality, the passive ones — interfaces and protocols —could be a basis for your implementation.

The first three chapters refer to some topics without going deeply into the matter. For these topics, Chapter 6 — *Supplemental Topics* provides additional background or explanation. Lastly, detailed comments about points in the main text can be found in the *Notes* section.

2. Overview of the Concepts

Terminology

To describe the third way, we will need a mixture of terminology drawn from information technology and also the social sciences such as human factors, management, social psychology, etc. A mixture of terms is unavoidable because this is an interdisciplinary boundary topic. Our objective lies in the domain of social affairs but the means lie solidly in the technology. To be sure that the following can be understood by everyone, I must explain a few technical terms.

Systems

A System is a collection of machines, networks, people, procedures, protocols, data standards, rules, databases, etc. It can be tricky to define the boundaries of a system because a system comprises many connected parts but a system also connects with many external parts. When is a connection between parts internal and when is it external? In order to explain our method for defining the boundary of a system, we need to examine what binds the components of a system.

The many parts of a system are bound together by ownership, control, and common standards. Ownership and control are often different. In real life a component part may be owned by one entity, it may be administered by another, and it may follow standards and conventions set by a third entity, e.g. a standards organization. If we want to define a system by finding a boundary, however, then it is clear that the standards don't define a boundary. Indeed, the purpose of the standard is to cover many and ideally all systems. Ownership is also unsuitable because an organization may own a system for one purpose, e.g. accounting, and another system for a different purpose, e.g. operating an on-line store. That leaves us only with one aspect to define the system boundary: administrative control. For our purposes, a system is an interconnected collection of component parts defined by its controlling authority.

Although any particular system lives within its isolated island of authority, the people responsible for the isolated systems usually recognize that their islands have interests in common and have areas where cooperation could be beneficial. But, the islands of control are also competitors and perhaps even enemies. Even enemies, however, will cooperate when threatened by another entity. Thus we need to build a society of systems to deal with the complexity of the real world. The real world has friends, enemies and some who are a little of both. Moreover, a system includes many people. Some people are reliable, others are dishonest or treasonous, and all will, on occasion, make mistakes. Although we rarely talk about it, a system has both character and reputation deriving from the actions taken by the administrative authority and the staff. The public character of a system, like that of a person, derives from the past history of how the system deals with other systems. The recognition, development and maintenance of reputation are important in any society and no less important to a society of systems.

A Society of Systems is any group of independently managed systems that communicate and work alongside each other. The individual systems in a society do not answer to a single authority hierarchy; therefore policies, standards, and goals differ and even conflict across the society.

This definition of *Society of Systems* is broad enough to include functional societies whose member systems work together successfully as well as dysfunctional ones that accomplish little. There is certainly a spectrum between functional and dysfunctional; moreover, a society can move towards functional on the spectrum when its member societies agree to work on a particular subject following specific data standards and communications protocols. It seems safe to predict that societies of systems will form gradually in small steps drawing together limited subgroups of systems. Let us call these limited subgroups *cliques of systems*.

It is possible to identify a few cliques that have formed already. For example, the many Domain Name Servers (DNS) that support URL lookup on the Internet are independent systems that have agreed to work together to support the Internet. Likewise, the Bitcoin servers that hold the block-chains have agreed to work together to support the Bitcoin exchange mechanism.

A Clique of Systems. The internet society is a voluntary association. Even when there is a control hierarchy as in the federal government, executive orders have very little impact on whether independent agencies work together or not. There are very solid reasons not to work together.

With the Third Way, there is a safe path and more reason to cooperate. On the other hand, not everyone will adopt the Third Way at first or perhaps ever. Instead, we expect that a few groups will agree to try it out and eventually more groups will adopt it. The systems that agree to work together constitute a *clique* within the Society of Systems.

The term clique is also useful because it has the flexibility to differentiate alliances according to their subject matter. For example, we might describe Europe as a multinational society but within it there are distinct subjects: economic, monetary, border control and defense. Not all nations cooperate on all subjects. For that reason, the society of European nations has arranged itself in four cliques: The Economic Union, the Euro Zone, the Schengen Area (countries that do not require passports to cross one another's borders), and NATO. There may be one Europe, but each European nation choses which clique or cliques it joins.

Encryption

Encryption is a technical capability that protects data from unauthorized use. Encryption is used throughout the processes and procedures of the Third Way; therefore, you should know several general points about encryption.

First of all, encryption is based on mathematical algorithms executed by a digital computer on data. The effect of encryption is that data becomes impossible to read or interpret (it becomes encrypted). In most cases, the encryption is reversible. That is, an algorithm executed by a computer on encrypted data restores the original, unencrypted data.

Secondly, the goal of encryption is usually secrecy so it is a surprising fact that the encryption algorithms are well known. But the encryption algorithm does not work alone. The algorithm requires an “encryption key” that must be supplied in addition to the data. Encryption provides secrecy only when the encryption key remains a secret. It follows logically that encrypted data are safe only if the data are kept securely separated from the key that was used to encrypt. As you learn more about the third way, you will realize that it is basically a method for keeping keys separate from the data while nevertheless permitting the use of the data in a productive but secure manner.

Third, it helps to know that there are two important categories of encryption algorithms: symmetric and asymmetric. A symmetric encryption algorithm uses the same key to encrypt or decrypt data. Asymmetric encryption uses a carefully crafted pair of keys. One key will encrypt the data while its partner will decrypt it. The asymmetric encryption algorithms were a major advance in cryptography because they allow the owner of the key pair to release one of the pair as a public key — a key that is known widely and verifiably associated with its owner. The owner keeps the other part of the key pair — the private key. This procedure, known as public key encryption or PKE, has amazing properties: authentication, non-repudiation, integrity, and confidentiality. There is no reason to explain these properties here. If you are interested, they are discussed in the supplemental chapter "Overview of Kerckhoff's Principle, Encryption, and Keys" on page 79.

Roles and Functions

Agents make things happen. An agent might be an actual person, an application operated by an actual person or a self-regulating process operating in the system on behalf of some authorized person. All the agents must assign a level of trust to other agents, even when the trust level is nil. More commonly, people trust the other members of their own group and certain software that their group controls. However, the society of systems brings together different groups and interests, so trust is a major concern.

Information Fiduciary. An information fiduciary is an agent operating in one system that operates on behalf of agents in other systems. The fiduciary represents other systems in their transactions within a society of systems.

For example, when you open a bank account, the bank is obligated to handle your financial transactions correctly. That is, the bank's fiduciary responsibility to you is its obligation to act on your behalf in certain transactions, like transferring funds when you write a check or pay a bill online or crediting the correct account when you deposit money. You can trust a bank almost always and the exceptions can be prosecuted under law. There is something special about a fiduciary relationship; namely, there is a contract or set of laws that bind the parties by obligations.

In a network of information systems, you are well advised to distrust the security of other systems and the motives of the various players operating in the network. Distrust, however, limits the value of the network for everyone. Looking to the future, we must introduce trusted fiduciary relationships throughout network operations before a vibrant, active and productive society of systems can evolve. I believe it possible to accelerate this process.

Blind Agent. The Third Way introduces a distinctive sort of fiduciary: the blind information fiduciary or blind agent. The blind agent's fiduciary responsibility is to handle encrypted content in a prescribed manner. The defining attribute of these special agents is their separation from encryption keys and unencrypted data. The blind agents are entrusted only with encrypted data that they cannot read without an encryption key.

Processes and Tasks

Stepwise Process. A process may be defined as: *A series of actions, changes, or functions bringing about a result.* Almost all software has a process with a series of steps. This book will walk you through a particular sequence that enables safe, secure information operations.

The sequence is distributed over many systems. We might say it is a process designed for a society of systems. Our sequence meets several desiderata:

- The steps of the process strictly separate all encrypted data from the key employed for encryption. Thus, an insider in one system cannot gain access to information in another system. This separation of encryption keys and data is a venerable and durable requirement known as Keckhoff's principle.
- Several steps of the stepwise process apply encryption using different keys for each step. As a consequence, data in transit is secured with at least three distinct encryption keys.
- The stepwise process establishes *need-to-know* before it permits information flow.
- The steps include various chokepoints that limit the amount and/or kind of flow. In view of insider attacks launched by employees such as Edward Snowden, Bradley Manning, and Aldrich Ames, any reasonable security plan recognizes that insiders may betray the trust placed in them due to the temptation of a financial reward or a loyalty to another cause. In addition, an insider may simply make a mistake that compromises security. Chokepoints limit the amount of damage an authorized insider can cause.
- All operations performed by the process honor stated restrictions imposed by the separate systems in a clique. The restrictions may include which partner systems are acceptable, what criteria are used to derive "need to know", and limitations on the quantity of information.

A Task is a job, process, or responsibility that is carried out somewhere in the system. We identify tasks so that we can assign them; moreover, some tasks are performed in sequence, some tasks repeat, and others may be occasioned by an event. The sequence and relationships of tasks over time is an important part of designing or understanding a system. The tasks will become important when we consider the implementation of the third-way operation.

Consent and Cooperation

The Third Way has the ambitious goal of building cliques of systems that voluntarily cooperate to achieve results. Each system is independent and can choose whether to join a clique or not. Moreover, having joined the clique, each system can choose to participate fully or in a restricted fashion. It is impossible to overemphasize the importance maintaining this control during operations because informed consent is a requirement for successful cooperation between independent systems. Overall, the Third Way recognizes several kinds of consent:

Other Members — When a clique has *m* members, each member agrees to participate in the clique with each of the other *m-1* members. The basic networking service does not ensure identification of any party engaged in a communication. Thus, an identity management system is necessary to ensure that we know who we are dealing with. During the formation of the clique, the membership in the clique should be revealed to all members and prospective members so that they make an informed decision about membership.

In an ideal world, everyone joining a clique would work together after joining. But it is not an ideal world.

Private Reservations — In public, clique members are committed to cooperation, but privately they frequently have reservations and wish to treat specific parties differently. Private suspicion is natural, given that each system is run by a different organization and different people. The Third Way must accommodate private reservations in order to foster the growth of cooperation in the community.

In practice, private reservations place a requirement on the Third Way; namely, any implementation must solicit consent for each actual transaction during operation. If this sounds onerous, please recall that this has happened from prehistoric times in the marketplace for things and services. Happily, complex markets operate successfully and we can expect a similar sophisticated market to develop for an automatic information infrastructure based on consent. The main obstacles to this type of informed consent are the demands from powerful members who want to control others unilaterally.

Acceptable Subjects and Procedures — Going back to the comparison with the marketplace, we can see that subject matter and operating procedures come into play. In our bazaar, can we conduct business for wheat or soy beans? What about slaves or opium? How long can we wait after a price is struck before the sale is final at the agreed price? If we look at natural social interactions, we see that clear rules are essential in a society and members must consent to them. Likewise, consent to rules about subjects and procedures is essential in a society of systems. To keep the discussion moving, I will simply note that this consent requirement can

Mediated Membership

We are all familiar with situations where people work together but the verification of identity is indirect. For example, at an auction of high-value art, many of the bidders may chose to remain anonymous because the successful buyer of the art object becomes a target for thieves. In this situation, people who participate in the auction constitute a clique but some of the members of the clique are buyers' agents. A buyer's agent must have a history of reliability and integrity so that the sellers can rely on agents to vet the actual buyer to be sure the actual buyer is financially qualified to consummate a sale. Thus, some participants may work through an authorized agent using an indirect identification system when such indirect verification is allowed by the rules and conventions of the clique.

be injected into a society of systems by the operation of the blind agent. Blind agent operation will be explained later.

Final Approval — When members of a clique have provided the three kinds of consent just described — clique membership, acceptance of the private reservations, and subject matter together with procedures — they have established the boundaries for possible transactions within the clique. However, the specific transactions are driven by events and the vast, rapidly changing information content available in the clique. Let us examine a transaction in more detail.

A transaction is an opportunity for two or more members of the clique to exchange related information that supports their individual objectives. Each transaction involves a small, carefully limited amount of information held by a few specific systems in the clique. If the transaction is executed, then the information is exchanged between all concerned parties. The approved procedures may be configured to allow the blind agent to execute the transaction. Then the final approval is moot. Assuming that the rules do not, we will discuss the final approval step.

Now, information for a transaction may take two forms. First there are facts and figures — obvious, incontrovertible data. Facts and figures may be the only data involved. The transaction may simply combine related facts to give each participant a more complete view of the situation. Second, there are specific queries asking for data that may or may not exist. In our view, queries are also transaction information. Many experts would chose, with some justice, to consider a query and its reply as belonging in separate categories. What combines the two for the Third Way is the heightened security concern. In fact, the specifics of a query on sensitive information are usually sensitive information as well. Consider, for example, an analyst seeking information on a private citizen who may or may not be involved with crime. The mere existence of the query imputes the reputation of the private citizen. Consequently, the analyst must take extreme care that the query does not become public. Moreover, if the private citizen is actually a criminal, the existence of the query would alert the criminal to take evasive action.

This dive into the interpretation of data analysis may strike you as a diversion but it brings us to a central matter. The search for connections between data and data and data and queries leads to highly specific, critically sensitive composite information. In the Third Way, this is nascent composite information at the point of discovery because the pieces of the composite are owned by different subsystems. Whether the composite becomes real hinges on the final approval from the systems that contribute to the transactions. Those systems in the clique that do not contribute do not even learn of the existence of the transaction.

No member of the clique can anticipate everything that might happen. Consequently, each member reserves the right to veto any specific transaction. A member might preapprove certain transactions but ultimately each member should have the opportunity to provide final approval.

These considerations may seem irrelevant from a technical perspective. However, a purely technical perspective is too narrow. If we expect software to support a society of systems, the consideration of social interactions is essential for eliciting cooperation from society members. Consequently, the requirements for consent and cooperation establish benchmarks for the software.

Information Fiduciaries

The distinctive feature of the Third Way's society of systems is the presence of a blind agent. As we defined it above the blind agent is a particular type of information fiduciary that handles encrypted information — encrypted information that it cannot read because it is isolated from the critical encryption key by physical and protocol barriers. The blind agent plays a specialized role in the Third Way but information fiduciaries can play many other roles in the society of systems. This chapter gives an overview of several important kinds of fiduciary system.

Fiduciary Responsibility

In the fields of law, business and economic theory, a fiduciary is defined as a party that is authorized to hold something of value on behalf of a client. A fiduciary has an obligation to handle the client's property with care and to remain loyal to the client's interests. The typical fiduciary has more than one client. You'll most often encounter a fiduciary when you are dealing with money.

Like money, information has value; consequently, the people who handle information must exercise fiduciary responsibility. Not surprisingly, we encounter a growing number of information fiduciaries all offering to hold and manage information for our benefit. However, there are several differences between money and information.

For one thing, money is fungible—it has the same value no matter who is paying whom. A dollar bill or euro note buys the same item in a given store, no matter who is making the purchase. The value of information, however, depends on who holds it. It's one thing for your best friend to have a photo of you acting a little crazy at a party, and quite another when a potential employer finds the photo on a social media site.

Moreover, money is a unique counter that can be counterfeited only with great difficulty, whereas information is easily copied. This issue has become a major point of contention since the advent of the Internet. The "information wants to be free" contingency and the people who earn their living creating, producing, and curating content have yet to come to an agreement.

So the two situations are dissimilar more than they are similar. Money can be stolen and used anywhere, and the victim of the theft always experiences this loss. On the other hand, information can be copied without loss to the owner. The damage to the owner depends on how the thief uses the information. Finally, even though theft is common for both money and information, everyone is rightly concerned about having their money stolen, but still today people show less concern for information.

A financial fiduciary or property agent is restrained by law and custom to apply care and maintain strict loyalty in all their operations. Information fiduciaries are new, custom has not been established, standards are not enforced, and legislation is only beginning to emerge. Everyone is moving into the information era with high expectations and perhaps unreasonable assumptions that information fiduciaries will act loyally and carefully.

For the moment however, we will leave such thinking aside and try to outline the patterns for information fiduciaries that will be important for describing the Third Way.

Custodian

In this section, we will explore the fiduciary aspects of systems by discussing three types of information fiduciaries. All three exhibit properties that make them custodians of data. A custodian guards, protects, and maintains things for clients. The fiduciary responsibility grows out of the custodian's possession of client property and the custodian's obligation to uphold the agreed upon terms of that possession. There are many complexities inherent in fiduciary responsibility that we explore here.

Computer science recognizes a certain type of custodial system called "client-server." A client initiates each action with a message and receives a reply back from the server. Most often today, you'll encounter servers that maintain a database of information (a collection of email messages, for example) that holds everything we or anyone else provides. When the client requests information (today's unread email messages, for example), the stored information on the server provides the substance for the server's replies to the client. One may also include the popular file servers in this category. A file server holds client files and returns a copy when the client requests one. Examples of file servers include Dropbox, OneDrive, and Google Drive.

You'll also recognize the client-server pattern when you contact your bank to check a balance or transfer funds. The pattern is that a client makes a request, the server takes an action and then the serve replies. In fact, the everyday exercise of standing in line for a bank teller and completing a transaction across a counter was once the standard illustration for client-server interactions. Today, the "teller" is often a computer that provides information through a website or phone app rather than through a glass window at a neighborhood bank. The fiduciary responsibility is obviously to accurately maintain all the financial records for the client. Less obviously, the responsibility involves positively identifying the client. It would be irresponsible to allow unknown parties to impersonate the client who owns the financial account data. It is the bank's fiduciary responsibility to properly identify clients.

When we go to a doctor's office, we are entrusting ourselves to a medical professional who has a legal requirement to keep our medical history in a server for electronic medical records. For example, mine are kept on two servers, one operated by John Hopkins University and the other by MedStar Health. These two servers are custodians of my medical information. Because the legislation that enabled electronic medical records was drafted in such a way as to encourage competition, the two servers won't talk to each other because the systems are incompatible. Like a bank these custodians make every effort to identify me and the doctors who use the medical record system. They keep the information safe. However, when medical records systems were envisioned, the utopian ideal for their fiduciary responsibility was that the custodians would

Clouds

At this point, an important development should be pointed out.

To keep matters simple, our discussion assumes each system does its own work. Expanding that statement using our earlier definition of a system, our assumption is that the work is done by component parts controlled by one authority.

More and more today, those authorities decide to outsource the work to cloud services. Cloud services are owned by large organizations like Amazon, Google, IBM, Microsoft, HP and others. It is reasonable to ask what fiduciary responsibility means in this contemporary configuration of components. You will find no answer in this place at this time. Cloud services are too new and the notion of cloud services will change as problems arise and are resolved.

In particular, the fiduciary responsibility of a cloud service provider is not well established by regulation or custom.

electronically maintain a unified description of the health state of the patient. It hasn't worked yet. A single patient is a separate client in each client-server system. Thus, I have one identity in John Hopkins and another in MedStar Health. Until those two systems cooperate in a kind of beneficial society of medical records systems, the objective of a unified health description for myself or other patients cannot be achieved.

Let's turn to the fiduciary responsibility of our custodians. Do they exercise care? Are they loyal to our interests?

Financial fiduciaries like banks are restricted by legislation to act in our interests, and long established custom reinforces trust. Moreover, we can choose what bank handles our transactions.

In the medical field, legislation has kept up with progress in information systems. There are laws imposing regulations on the care and use of medical records. The paper work associated with the compliance with regulations may be onerous and the enforcement of regulations on the systems may need improvement but the legal framework has been established in the United States and many other countries. At this time, a patient has no choice, the custodian for medical records is selected by the doctor.

These two fields, financial and medical, serve to illustrate the issues involving a system's fiduciary responsibility. Other fields could be discussed but two will be sufficient at this time.

Curator

A curator acquires and maintains collections of information from multiple sources. A curated information system is like an art museum where experts collect and display objects for visitors. In a similar way, visitors go to a curated site to find and view information.

It is possible to regard the curator as a subclass of the custodian. A curator acts as a custodian but additionally adds value to the data by annotation, indexing, cross-referencing, and time-stamping.

The best known curator today is Google. It finds and reviews as much as possible from the Internet and penetrates deep into other online collections, e.g., it indexes the PubMed web site at the National Library of Medicine as well the health information on WebMD and the videos on YouTube. All of that curated information is prioritized and made available to clients over the Internet. Likewise, Google keeps track of its clients' activity and derives information about the clients—another collection Google maintains with less publicity.

Most people would not classify Google or any other web curator as an information fiduciary; but, perhaps they should. The information Google collects may be free but the information derived through curation and display is novel, valuable, proprietary to Google, and widely sold. Sometimes that sale of proprietary information benefits Google's clients. For example, when I visit a travel website, the site is likely to know already where I like to travel because they've paid for information about me derived from my Google searches. Naturally, they tailor their offers to my interests that are revealed by Google's knowledge of my previous searches. Generally speaking, that works for me.

What else happens to Google data? Is it handled carefully and in the interests of its clients? That situation is murky. Clearly, there is no expectation that today's information curators will be loyal to anything other than corporate interest.

Matchmaker - Exchange Markets

The electronic exchange market serves as a fiduciary for information about buyers, sellers, and the property they can exchange usually for a price. In a sense, an exchange is a specialized information fiduciary that is focused on arranging pairwise transactions between a buyer and a seller. By now, all stock exchanges in the developed world have become electronic exchanges. In addition, there are successful electronic property exchanges like eBay.

All of these exchanges have their rules for trading, which they publish. We expect them to exhibit a fiduciary responsibility to negotiate prices and sales according the published rules. But, in spite of best intentions, the exchange market may allow a customer to come to harm. For example, sophisticated users of eBay have found bidding strategies that give them an advantage over casual users. Likewise, high speed stock trading firms have learned to turn a profit by investing in short-delay access to a stock exchange and then exploiting transient market inefficiencies.

The Third Way supports a new specialization of the exchange: encrypted exchanges. Few examples exist. However, an encrypted exchange can be useful when the participants in an exchange want to arrange transactions among themselves without revealing to the public the terms of the transaction such as price, quantity, quality, or other attributes. We will see proposed examples of encrypted exchanges in *Part 3 — Applications.*

Mediator

There are several examples of an important pattern we call the mediator. A mediator offers a service involving more than one party. Here are two examples.

When you look for a low-cost airfare using Travelocity.com or similar service, you are using a mediator that aggregates data from many sites, prioritizes them by price, and offers the user the opportunity to choose which ticket to purchase. Likewise, when one shops Amazon.com, one sees what Amazon offers at what price but, Amazon may also offer you an opportunity to buy from a third party.

These examples do not exhibit a fiduciary responsibility. There is no expectation of loyalty or care using Travelocity or Amazon. We employ the mediator for our convenience and they take their profits directly or by an arranged fee from the third party who provides the product or service that we purchase.

To be sure, there is one aspect of a sales provider that involves fiduciary responsibility. The sales provider learns our personal information and credit card number. For that, they have a fiduciary responsibility. By and large however, we use these providers for convenience and don't worry about our information.

For the Third Way to work, it must incorporate an essential mediator pattern whose operation is predicated on trust and fiduciary responsibility. That mediator is the certificate authority or identity service—an essential service wherever trust and responsibility matter. At a fundamental level, fiduciary responsibility is what one party owes another. But who is who? Who owes what to whom? Trade and business falter unless there is a positive mechanism to ensure identity. Otherwise, can we can never speak meaningfully of fiduciary responsibility.

There is much more to say about certificate authorities and identity management systems but the subject is too broad to pursue here. We take for granted that an identity authority exists and works.

Remote Agent Host

In the past, when networking between computers was novel, people assumed that software agents would run on a local system. Later, there was a flurry of interest in agents that could travel around the Internet like a computer virus and do work at remote locations. The idea was that the agents could find results in large databases and return them without the overhead of moving data to a central location. Needless to say, this kind of remote agent was far too much like a virus, and most remote agent systems never escaped the prototyping phase.

In certain cases, however, a remote agent is unavoidable. If I want to operate a Mars rover from a comfortable control center on Earth, then you must deal with limited bandwidth and long delay. Only remote software agents can perform the minute-by-minute operations at a site as remote as Mars. In this case, however, the agent software has been carefully vetted. Changes to the remote software are few and tightly controlled. Only the specific parameters and to-do lists are updated from day to day. Because there is tight administrative control, the space explorers at NASA enjoy the advantages of remote, “situated” agents without the threat of computer virus infections.

Another example occurs in cloud computing wherein clusters of computers are martialed to complete a large job. A job is decomposed into pieces that are parceled out and handled by agents assigned to many individual computers. Each individual computer serves as a host for the remote agents. This approach is called sometimes “MapReduce”. (We are admittedly spinning the interpretation to say it is an example of remote agents.)

It is not necessary to deal with remote agents to implement many, if not most, applications of the Third Way because the basic set of encrypted matching operations will be sufficient to evaluate a match. However, it is easy to think of applications where that is not the case, such as matching a gene sequence to a protein sequence in studies of DNA[ii]. In the protein/DNA example the technical issue is that several gene sequences can match the same protein sequence. The correspondence of protein and DNA is easily recognized in the unencrypted protein and DNA sequence data but impossible with the encrypted sequence data. Therefore, a conversion must be performed by an agent in the location of unencrypted data before any data are encrypted and sent to the blind agent for encrypted matching.

Fiduciary Responsibility in a Society of Systems

All of the traditional types of fiduciary appear now on the Internet or show a potential for future applications. Yet there are barriers to their acceptance. The first is lack of tradition. We have all had experience with banking but not so much with on-line health services. This barrier will fall as we gain familiarity with internet services. However, any applications that involve sharing sensitive or private data face a permanent barrier intrinsic to information sharing.

My information has equal value whether you acquire the original or a copy. In fact, you can’t distinguish the original from the copy. In practice, that means if I want to keep information to myself, I cannot even show it to you because information is easily copied. The fact that information is an arrangement of matter but itself immaterial makes it easy to steal any valuable information. The victim keeps the original but loses the value.

It is so easy to steal valuable information that anyone who holds information must keep it completely out of view. For that reason, there is no way to easily share it with another for mutual benefit. As a result, the society of systems requires a new fiduciary called the *Blind Information Fiduciary*. This new fiduciary is, perhaps, the most distinctive feature of the Third Way. We turn to that subject next.

The Third Way: Use a Blind Agent Information Fiduciary

Now that we have discussed the importance of fiduciary responsibility and the pervasive element of distrust in a naïve population of systems, let's apply this concept to third-way software technology. We need to show that software technology can change — not in any earth shaking way but using only small additions of code — so that distrust is pushed aside allowing our naïve population of systems to evolve into a functional, effective society of systems. Let’s explain the software notion by walking through a specific situation.

Intelligence agencies with antiterrorism responsibilities are divided by administrative fiefdom and by location across international boundaries. Too often intelligence agencies retrospectively assemble evidence that could predict a terror incident by searching across multiple information systems, but the search usually happens after an attack to apportion blame.

One example is the incident of the "underwear bomber" Umar Farouk Abdulmutallab. An after-the-fact analysis revealed that before he boarded a commercial flight in December 2009, Abdulmutallab was mentioned in four systems located in three countries: the US, the UK, and Yemen. In Yemen, Abdulmutallab was seen studying at the radical madrassa run by the American imam Anwar al-Awlak — a mentor to many terrorists. In the UK, his entry visa was suspended because of his association with terrorist cells in Britain. In the US, he was on the CIA’s list of suspects because his family in Nigeria had reported his radicalization and potential for trouble. Abdulmutallab also held a visa to enter the United States freely, and that fact would have been stored on a US State Department computer. Yet Abdulmutallab was not placed on a watch list in the US. Consequently, he was allowed to board an airliner freely. The potential victims on the airplane were spared destruction only by the faulty chemistry

Cancelled
Visa list
Multiple-entry
Visa list
Terrorist suspect
list
List of students
Of radical Imam

employed in the detonator for Abdulmutallab’s explosives. It was a near miss, one that might have been avoided entirely.

Why can't intelligence agencies perform the same analysis before an incident that they perform retrospectively? That proactive analysis would have prevented this particular attack. To be sure, agencies are working on this deficiency. However, there is a fundamental and even essential distrust that exists between agencies, and this creates disharmony. We might cite many reasons for this distrust, but for the current discussion we will focus only on one.

Suppose we have two systems, each of which supports an independent intelligence agency. Let's call the systems Wintermute and Neuromancer (from the novel "Neuromancer" by William Gibson). Because the two systems are independent, each follows its own rules and procedures. They pursue different agendas and each is surrounded by human actors who act independently, unreliably, and occasionally unfaithfully. In addition, the two systems may need to maintain separate information archives as a legal requirement — as was the case for the fictional Wintermute and Neuromancer. Thus, the two intelligence agencies naturally distrust each other for many reasons: the other's employees are strangers, the other's managers are bumbling bureaucrats, the other country has laws that undermine our system's integrity, etc. Is it any wonder the agencies' systems can't get it together to cooperate and stop the terrorism pattern before it is instantiated?

Today, the widely deployed software technology adopts a version of the federated database where information is stored in separated information domains joined by network connections. In operation, predictive software must be executed by an agent that works within one system but draws data from other systems over a secure conduit as illustrated in Figure 1. In this illustration, an agent running in Neuromancer discovers a connection or linkage between facts that derive from two sources, Wintermute and Neuromancer. That is the power of federation: pulling facts together to create results. The weakness is also obvious. Federation amalgamates the boundaries of all the systems, creating a much larger security perimeter with many more points of weakness. Which system is weakest? The assumption is that it is always the other guy. Why trust what you can't review, correct, and, if need be, punish?

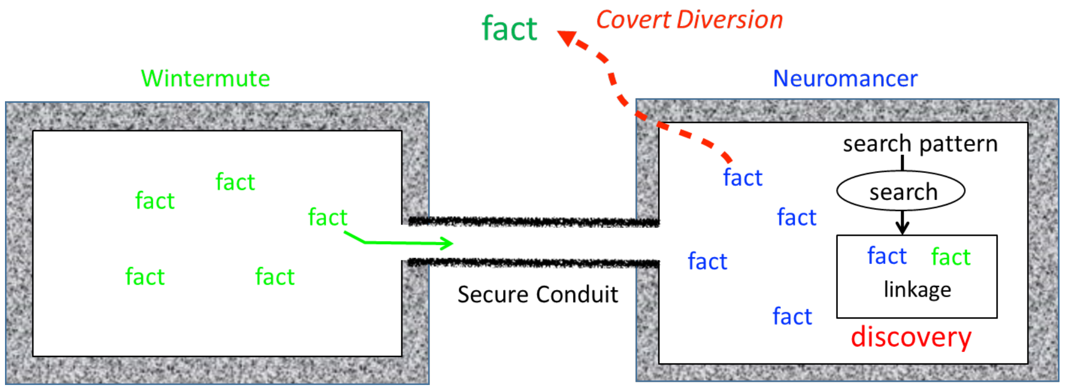


Figure 1 Conventional Cooperation between Two Systems

In Figure 1, there is a security leak in Neuromancer that diverts information that administratively belongs to Wintermute. Consequently, Wintermute fails in its fiduciary responsibility to its owners because it cooperated with Neuromancer, allowing data entrusted to Wintermute to be diverted from Neuromancer. The administrators of Wintermute know this data loss is possible, so they are rather unlikely to cooperate fully with Neuromancer no matter what memoranda of understanding or Executive Orders dictate cooperation.

Without cooperation, there is no discovery, facts are never put together and we do not "connect the dots." That is the current state of affairs. The state is one of failure, failure that we must attribute to the decision-makers who authorized the system architecture. People who have the control of systems failed to organize the information system for cooperation. That may have been acceptable in the past. Today and tomorrow, the Third Way improves the computer functionality so that machines foster human cooperation without compromising security.

Third-Way Approach — It is clear that the actual discovery operation cannot reside in either Wintermute or Neruomancer for security reasons. So the Third Way introduces a separate system, we'll call it Zatoichi[3], that can house the discovery agent away from either Wintermute or Neuromancer. This works as long as you can trust the security of the new system. In short, you can't — each system is a potential point of failure.

The Third Way solves this problem by restricting the intermediate agent, Zatoichi, to use only encrypted data. Then, if the security of Zatoichi is compromised, the attacker gains only encrypted, useless information. That is the basic notion of a blind information fiduciary. The Third Way unites encrypted information to achieve the results of a federated system with the safety of compartmentalized security.

The blind information fiduciary is illustrated in Figure 2. There are major configuration changes for Neuromancer and Wintermute as well. Each sets up internal security barriers to separate protected information from encrypted copies of the information. The configuration of Figure 2 removes the discovery agent that was running in Figure 1 on Neuromancer. It relocates the discovery agent to the

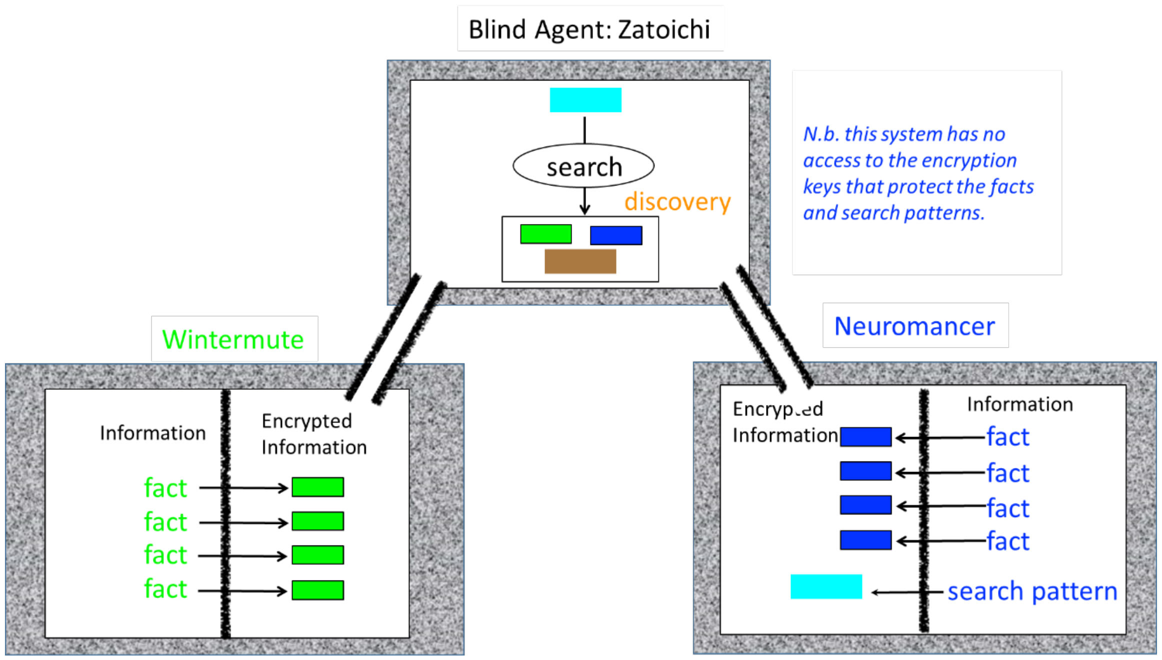


Figure 2 Safe, Secure Information Integration through a Blind Agent Information Fiduciary

[3] Zatoichi, the fictional character created by novelist Kan Shimozawa, is blind, but nevertheless skilled and effective.

blind information fiduciary, Zatoichi. After the agent moves to Zatoichi, the agent still needs to know what linkages to look for, and those goals are selected by the agent's clients. Figure 2 illustrates this agent-initialization by showing that Neuromancer originates a search pattern, which is also encrypted[iii] and sent to the agent.

Returning to our motivating example on page 22 regarding the underwear bomber incident, let us imagine that the blind agent operates on an international scale. It would then be possible to connect agencies across national boundaries and predict more events before they occur. So why isn't that a normal procedure?

All the best intelligence systems in the world cannot prevent trouble while we dodge the real issue: allies and friends are all mixed up, we can't keep them straight, and so we tell our computer systems to distrust. Why, in this example, would "allies" distrust each other? Here are a few reasons.

- Reports can be tracked to their source by the circumstances detailed in the report. In Yemen, there will never be more than a few sources, perhaps not even one. Thus, each source is valuable and must be protected from exposure. Because sources are valuable, access to source-sensitive information has to be heavily restricted.
- Reciprocity with Yemen is unlikely because the United States cannot trust all the state actors.
- Reciprocity between the UK and the US is compromised by a history of intelligence failures on both sides, court rulings seen as unfavorable on the other side of the Atlantic, and arbitrary action, e.g. the outing of CIA agent Valarie Plame for political revenge.
- For agencies within one country, say between the CIA and State Department, cooperation is impeded by differing objectives and strategies.

How can the Third Way change that situation? The Third Way protects by preventing any party from obtaining free access to information held by another party unless several conditions are met: the other party approves the discovery method in advance, the other party is notified of every discovery, and the discovery provides evidence that there is a significant connection between specific items held by the parties — in other words, there is a "need to know". At the same time, the Third Way enables intentional, well-considered sharing because it discovers "need to know" effectively using a synoptic view of federated, encrypted data.

To summarize, because there is good reason to distrust other parties, a blind agent identifies specific opportunities to exchange data and cooperate. There is still a risk, but the risk applies to a small amount of data. Unlike current federated systems, the entire collection is never at risk to an insider or an intruder. Thus, we can expect better cooperation.

Use Cases

Purpose. Our overview thus far has placed the Third Way in the context of computer science; furthermore, we pointed out the distinctive third-way features. But what actually goes on in a society of systems? What stories can we tell about it? To answer such questions, we turn now to “use cases”. The “use case” is a term adopted by a currently popular method for documenting system design.

To paraphrase Ivar Jacobson[4]: *A use case is a way of using a system to achieve a particular goal for a particular user. Taken together the set of all the use cases gives you all of the useful ways to use the system, and illustrates the value that it will provide.*

Each use case is a short sequence of back and forth interactions between actors and system elements. The sequence explores part of a story line. A small set of use cases can explain a small application but a modern enterprise system requires so many use cases and so many stories that it should be compared to a saga like *Lord of the Rings*[5] or *The Sopranos*[6]. Fortunately, we only need a few cases to provide an overview.

The use case viewpoint captures the goals, expectations, and rationale for a system. Moreover, use cases also form a bridge between the technical and non-technical communities. Each use case has a story, some pictures and some additional features. Between the story and the pictures, most everyone will figure out what the case portrays because stories are a very natural communication medium[7] and pictures are rich in meaning.

Operations. If you remember from earlier that we define a system by its independence from other systems, it should not be a surprise to say that the staff of one system always use that system and that system alone. Of course, they may receive information from other systems. Indeed, the point of introducing the Third Way is to make sure that they learn what they need to know from others via a safe mechanism. But when the Third Way is taken, we can separate the concerns about operations as follows.

- ❖ Users on the staff with one system interact with their own system, not other systems. See the illustration in Figure 3 on page 27.
- ❖ One system interacts with other systems in the society of systems but not with the users of the other systems. See the illustration in Figure 4 on page 27 .

For the moment, we will focus on the users and then consider the interaction of a single system and the society of systems in the next section, *Sessions*.

[4] See “Use Case 2.0” by Ivar Jacobson, https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf
To be honest, I have not paraphrased but rather modified the quotation from Jacobson for clarity.
[5] *The Lord of the Rings* is a trilogy of fantasy novels written by J. R. R. Tolkien and released in 1954.
[6] See http://www.hbo.com/the-sopranos
[7] See for example “The Storytelling Animal: How Stories Make Us Human” by Jonathan Gottschall, 2012.

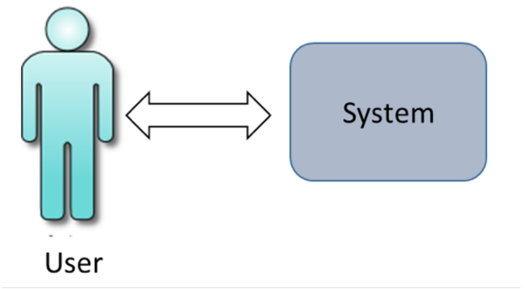


Figure 3. The User Interacts with the User's System

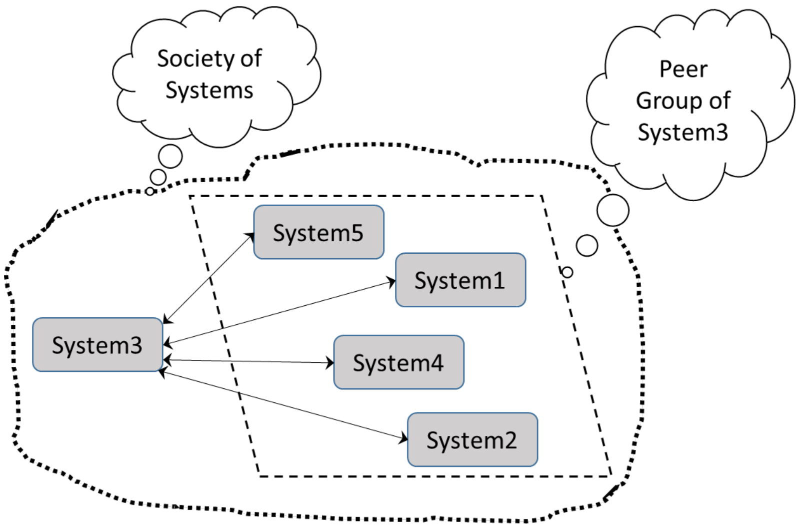


Figure 4. System3 Interacts with its Peers in the Society of Systems

Use Cases for User Operations. We will focus on the essential use cases that define and differentiate the third-way system. Here are the main ones:

1. User makes a specific information request expecting an immediate response and receives it after a short latency time. See Figure 5 on page 28.

2. User submits a long-duration request expecting to be notified whenever information is available that satisfies the request. See Figure 6 on page 28

3. The user receives results and responses that are not related to the user's current focus. Moreover, the responses arrive at unpredictable times. Each response is annotated with the context of the event that precipitates it. On that basis, there are two sub-cases:

    a. The context is a request that was submitted previously that is persistent in the system and that presents new results when new information appears somewhere in the clique of systems.

    b. The context is a request submitted by another user who designates this user in a distribution list.

We are all aware that a user has many different needs and demands. The previous three are just the use cases that are central to our explanation of the concepts. Many other important use cases can be safely ignored here such as these:

- User updates or adds information to the system.

- User stores and organizes results received in a response from the system.

- One user communicates with another regarding a request or a response.

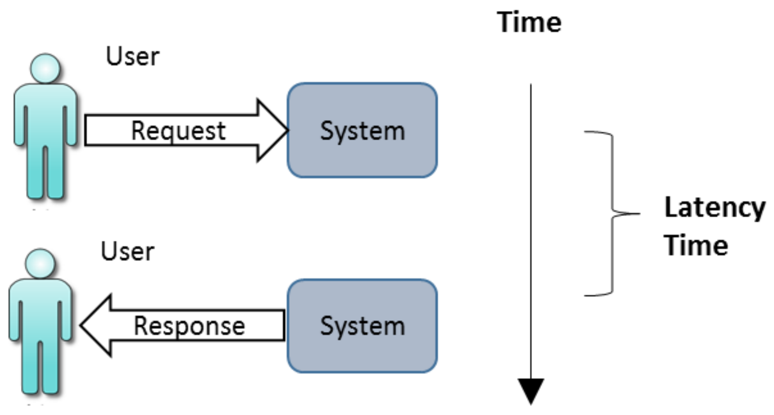


Figure 5.Use Case 1.

Let's look at the first use case with the help of Figure 5. The use case is extremely simple and it represents the ideal situation for most users. They want information fast and in this case, they get it almost immediately. Inevitably there is a latency time before results come back. The latency time is often a critical factor in a user's evaluation of the system. Now, the fact that the user expects the response is critical in understanding the story. The user is doing something they control and the system responds to that control. At no point does the user need to worry why the system is sending the response. Remember, the use case is ideal. We've all used a system that made a response so unexpected that we have no idea what it was doing. Ideally however, the user always knows the contextual information around the response because the user initiated the process very recently.

Apart from the lucky few who have easy jobs that just need a simple request/response use case, users in general need to wait. There are many factors that force a delayed response:

- a lengthy computation is required,
- information is required from backup storage and the storage media needs to wait in a queue to be mounted on a device,
- the request involves dynamic information that triggers a series of responses as new information arrives and lastly
- the user needs help from others.

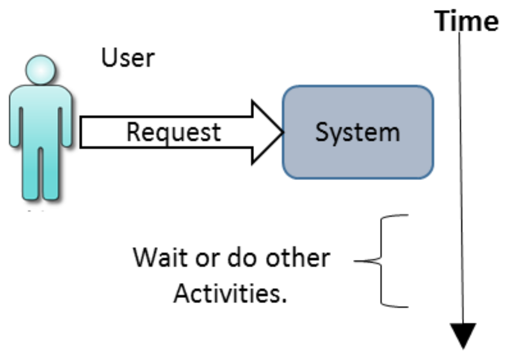


Figure 6. Use Case 2. Submit a request with long or unpredictable waiting time.

The "help from others" may need more discussion. Use Case 1 implies that a user has all the access permissions and ownership rights to get back the desired response. In many situations, however, the user lacks permission or ownership and must ask for permission. In the past, the user would file paperwork, make phone calls, seek permission, maybe exchange a few favors, etc. In the future, the process will become fast, reliable, and secure by adoption of the Third Way. In whatever way it is done, there is a clear requirement for Use Case 2. Use Case 2 is illustrated in Figure 6 where we see the user starting a request with the system.

During the course of a day, a user may follow Use Case 2 repeatedly to start work in the system. At the same time, a user's colleagues will be hard at work as well. The work flow in an organization involves a lot of collaboration so information doesn't necessarily go to one person; it goes to many. The best model for working at a job today is the email model. We stare at a screen and messages, pictures, attachments from dozens or hundreds of conversation threads pop into the inbox. The typical user today is event driven or as techies say “interrupt driven.” Now for a techie, an interrupt is a hardware concept but it works as a metaphor as well. The typical worker is interrupted constantly. The Third Way has absolutely nothing to say about the productivity impact or cognitive attention issues that follow from driving people with events. Unfortunately, the Third Way entangles us more deeply in the interrupt condition as we can see from Use Case 3.

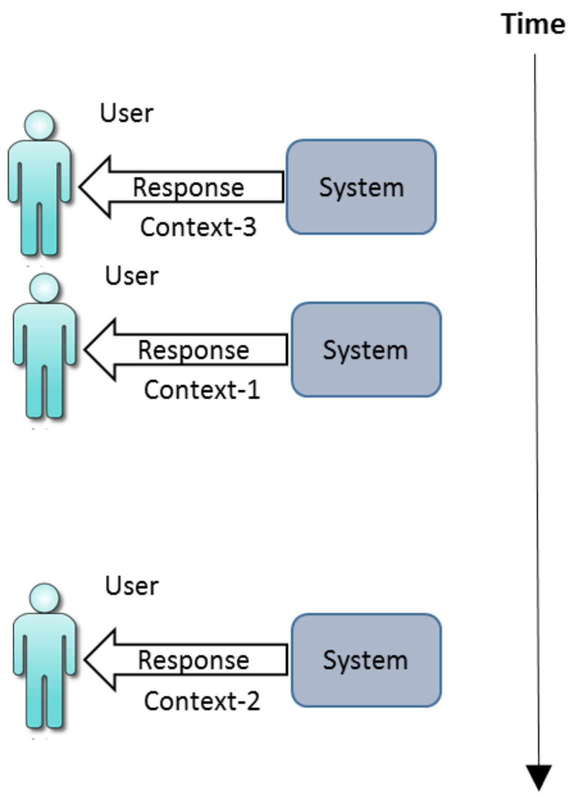


Figure 7. Use Case 3 - User receives responses associated with different contexts and arriving at unpredictable times.

In Use Case 3, Figure 7, responses come to the user in unpredictable order, at unexpected times, from one, two, several or many simultaneously active processes in the system. While not ideal, a system must deal with this use case.

Since this is a conceptual overview, we don’t say much about developing software for a successful product. However, if that is your business, you’ll find Use Cases 1 and 2 are fairly easy. You can build an interface to the system for the user and let them use it. Frankly, our final case, Use Case 3, is in a whole other realm of difficulty. The user receives sporadic information from several contexts. The key to the success of the product and the factor that users will judge most harshly is how context is employed to organize and present the results. Again, this is an implementation consideration so we will say little about it. But it is worth noting that it calls our attention again to the fact that in a system, and even more in a society of systems, a user is dealing with many simultaneously active agents. Simultaneous agents represent a frontier area for machines and people. It is fair to say, however, that people cannot focus well on multiple intellectual tasks simultaneously. Thus, users need a lot of help handling their job in Use Case 3.

In the following section, *Sessions*, we will discuss an additional Use Case that covers interactions between Systems as illustrated in Figure 4 on page 27.

Sessions

The last section of our overview presented three use cases that describe the interaction of a user with a system. Those use cases are common and well known. Now we turn to less common, new use cases — use cases that describe how one system interacts with other systems in a society. Figure 4 on page 27 showed the high level use case and illustrated how interactions are really among a clique of systems that have mutually agreed to work together. The Third Way eases the way for cooperation among the peer systems — systems that are independent and treated equally during third-way operations. Some systems serve a fiduciary function and are not peers. For example, there must be a system that all the peers agree is an authoritative source of identity information. Such a source participates in several essential use cases:

- One system establishes its identity with the identity information fiduciary.
- One system verifies the identity of a second system by consulting the identity information fiduciary.

There is no reason to discuss these use cases here — just be aware that identity services are an essential aspect of secure systems. Often, these services are provided by so-called *Certificate Authorities*. The identity services are well understood but complex. [iv]

There may be other non-peer systems such as the *Adjudicator* discussed in the *Supplemental Topics*. Here, however, we must discuss the one non-peer system that is essential — the *blind information fiduciary*. To understand the use cases that involve this fiduciary, we need to introduce the concept of a *session*.

The conventional meaning of a *session* is a meeting of people for a purpose, at a time, and for a certain length of time. The third-way definition is very similar. According to the Third Way's definition, a session includes these essential attributes:

1. A session involves a defined set of member systems — the clique of systems.
2. A session starts at a time and lasts for a certain interval of time.
3. Ideally, a session produces results for its member systems.
4. Each system may engage in multiple, simultaneous sessions; therefore, it is important that each session should have a unique identifier to distinguish its properties, context, and state of progress.

These first four essential attributes are fairly straightforward. Now our session gets specialized for the Third Way:

5. A third-way session involves members who are peers plus a blind information fiduciary system that is a member of the clique but a member on unequal terms. This fiduciary is not a peer. This distinction is central to the Third Way.
6. A third-way session has an associated session encryption key that is shared by peers but this session key is never shared with the blind information fiduciary. The fiduciary is blind because it

does not have the encryption key and therefore cannot decode any private or sensitive information.

Earlier, we examined the blind information fiduciary in Figure 2 on page 24. Now we discuss the operation within an extended explanation of Use Case 4. For convenience, we illustrate the single use case with a sequence of figures beginning with Figure 8. In Figure 8, we see the start of session. There

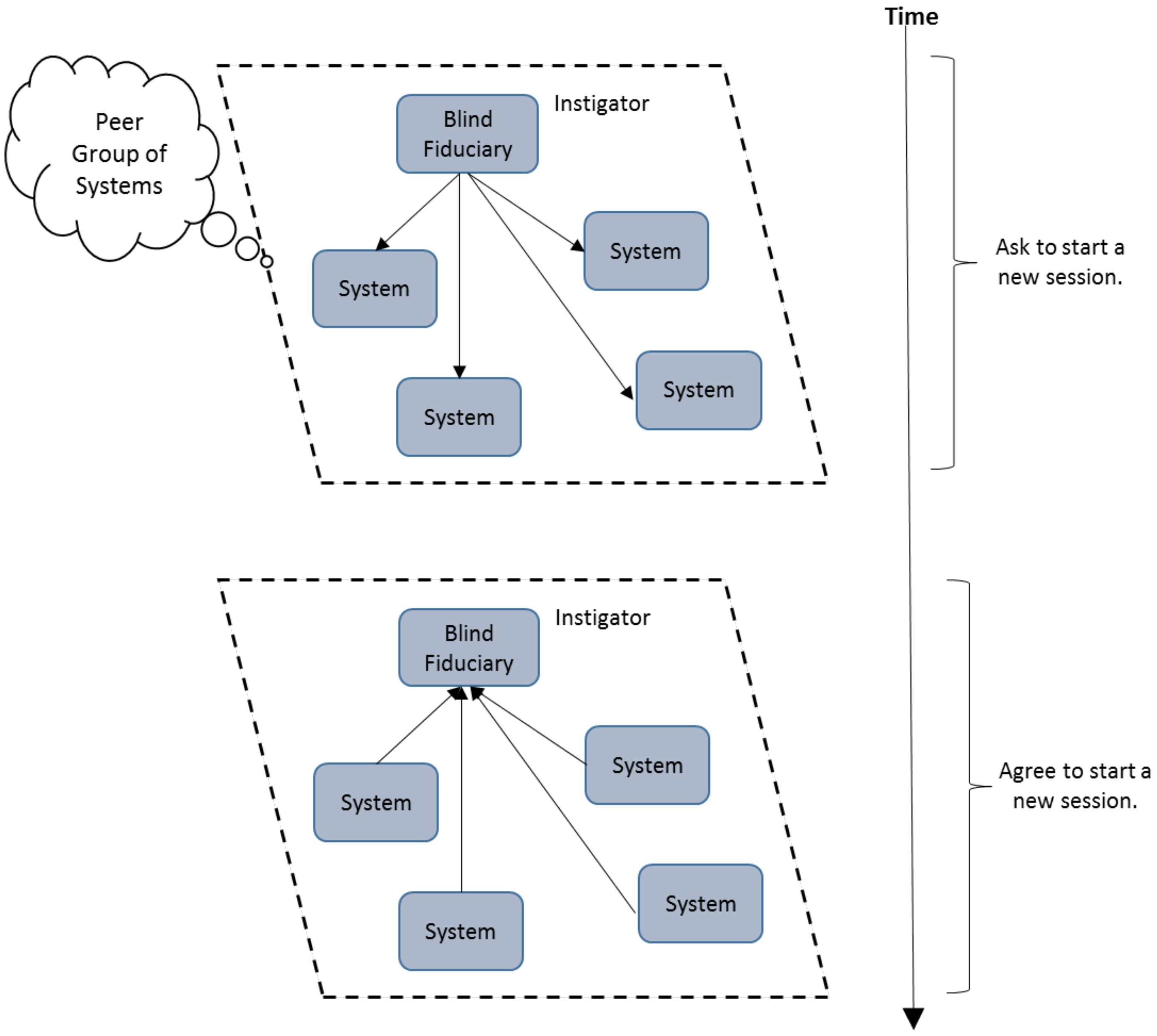


Figure 8 Use Case 4, Part 1, Start of a Session

are many events that might start a session. One of the peer systems might request a session. The session might start on a prearranged schedule. Or, as we have shown here, the blind information fiduciary starts the session. The first step is to propose a session with specific properties and rules and a

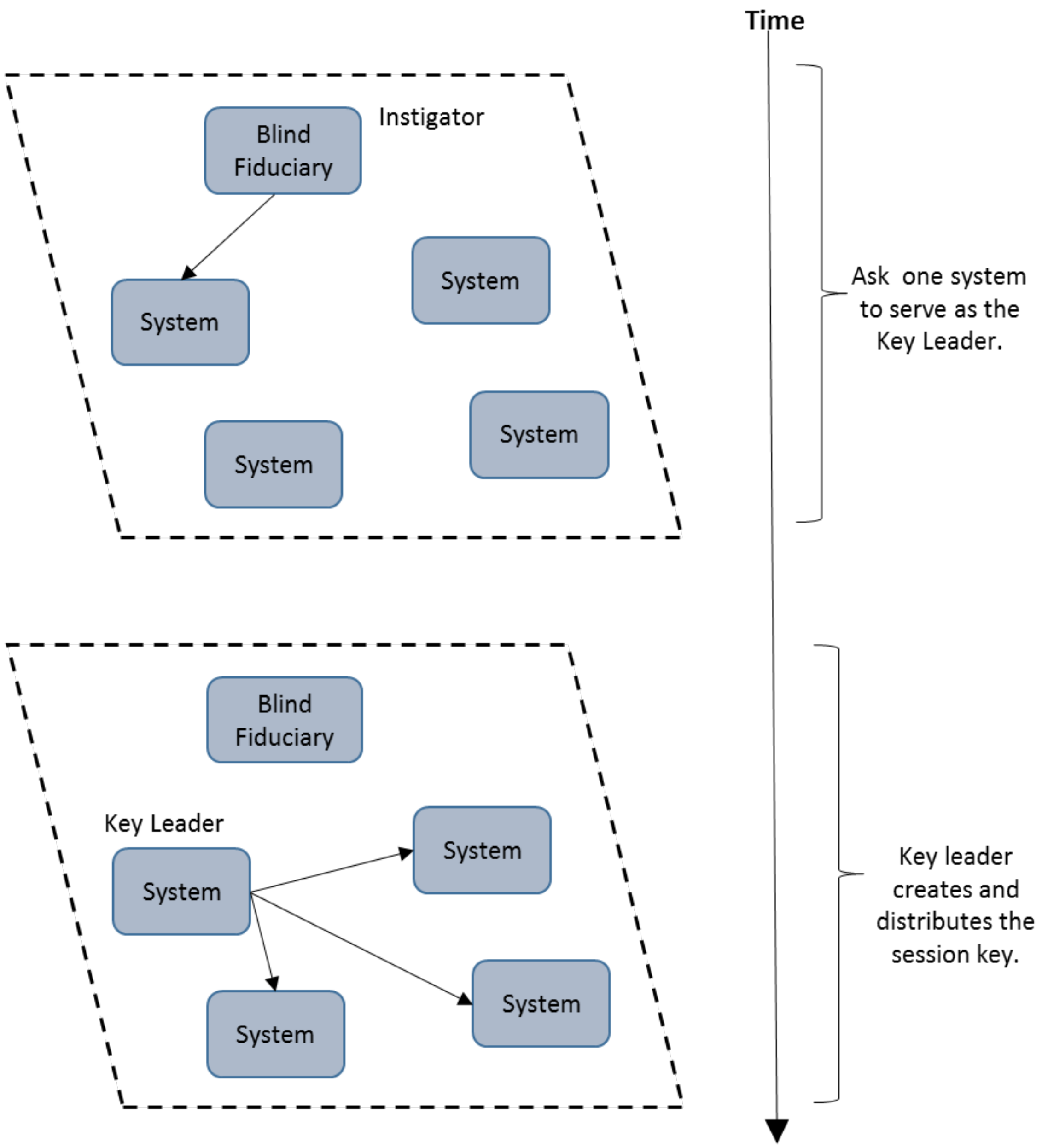


Figure 9 Use Case 4, Part 2, Distribution of Session Key

unique identifier. The instigator must ask for concurrence from all the peer systems in the clique and then wait to receive their concurrence. If the instigator cannot obtain approval from all members, it may either try again later or it may decide to create a subset clique of those peers who have consented.

In the next part of the use case, one of the peers creates a unique session encryption key and distributes it to the other peers that have agreed to participate in the session. We'll call that special peer the *key-leader*. If the systems are peers, which one should be special and become the key-leader? If one of the peers instigates the session, then that peer is the obvious choice for key-leader. However, in our illustration, the blind fiduciary was the instigator. Of course, the fiduciary cannot serve as key-leader because the whole point of the game is to prevent the fiduciary from seeing the key. What happens then is illustrated in Figure 9. Here we see the blind fiduciary picks one of the peers to serve as the key-leader. The key-leader then distributes the session encryption key to the other peers but not to the blind information fiduciary.

During the next phase of the use case, the individual peer systems decide what data they are willing to contribute in encrypted form to support the goals of the session. Presumably, each peer system has built a protected area isolated from its secure storage and has carefully populated that protected area with encrypted information (See Figure 2 on page 24). When all the information is ready, the peer sends it to the blind information fiduciary which federates or combines the data. This phase is illustrated in the upper part of Figure 10. When the encrypted information becomes available, the fiduciary begins processing the encrypted data to discover connections or relationships that are new or novel in

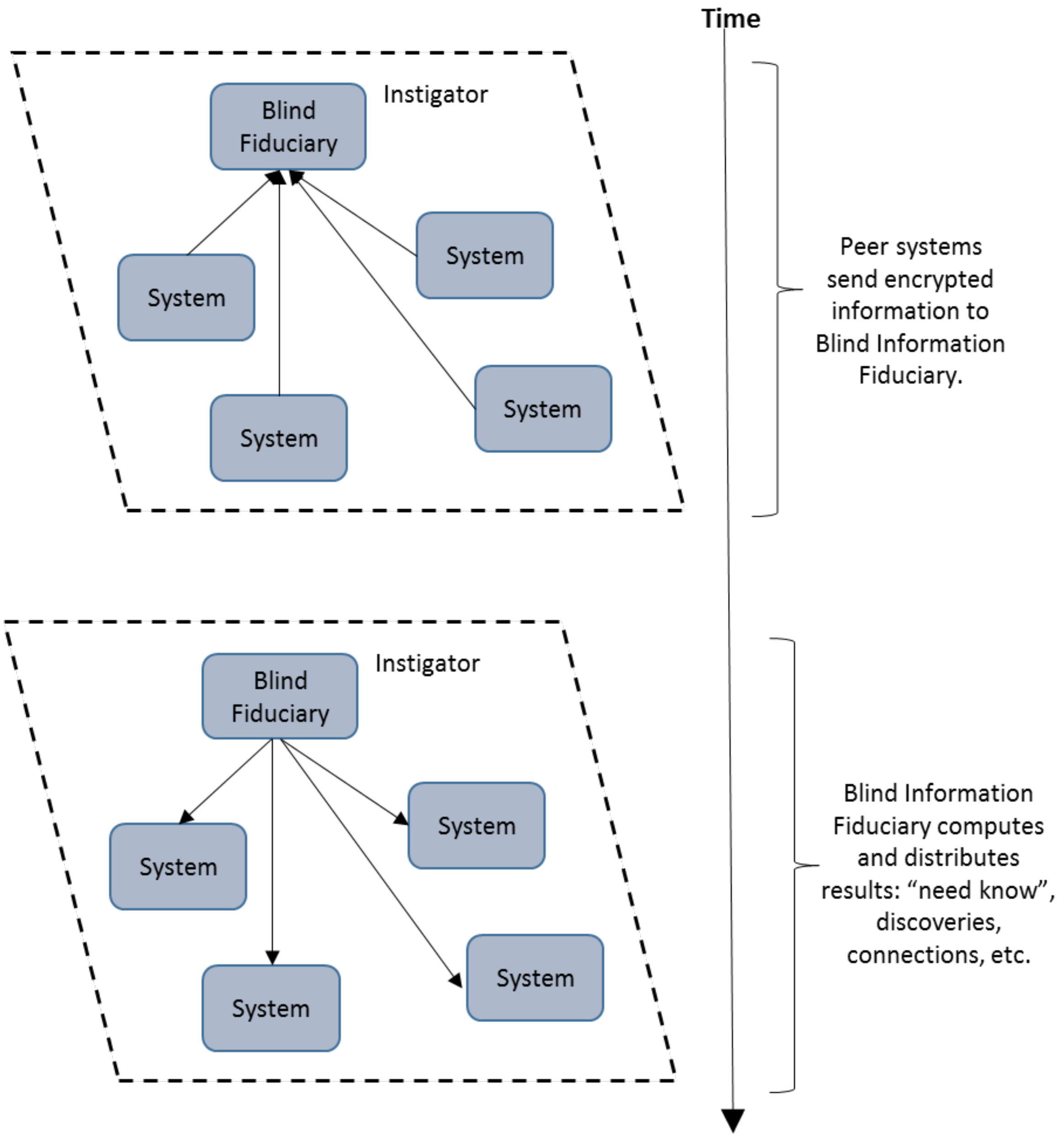

Time
Blind Fiduciary
Instigator
System
System
System
System
Peer systems send encrypted information to Blind Information Fiduciary.
Blind Fiduciary
Instigator
System
System
System
System
Blind Information Fiduciary computes and distributes results: “need know”, discoveries, connections, etc.

Figure 10 Use Case 4, Part 3, Matching and Discovery from Encrypted Information

themselves or may function as need-to-know justification for subsequent sharing data among the peers. The fiduciary may also calculate encrypted results considered important by the clique, although the possibilities for numerical calculation on encrypted data are fairly limited. In any case, this part of the use case concludes when the blind information fiduciary distributes encrypted results to the peers in the clique as illustrated in the lower part of Figure 10.

The results computed by the blind fiduciary are encrypted as a natural and unavoidable consequence of the fact the results are derived from encrypted information. Although the blind fiduciary that computes the results cannot read them, the results are meaningful for the peers of the clique because peers have all adopted the same symmetric session encryption key. Therefore, peers are able to decrypt the results.

After the Blind Fiduciary has distributed results, it cannot call them back. In many cases, the members of the clique will consider that outcome efficient and reasonable. More likely, they will want more selectivity in the distribution. Consequently, the rules of each session must tell the fiduciary how it should handle its discovered results. Here are several possible rules:

1. Return a result to a peer system if and only if that system contributed information leading to the result. This rule prevents distribution of results to peers that have not demonstrated a need to know by exhibiting some knowledge of the matter at hand. This rule is strongly recommended for every session in any peer group.
2. The blind information fiduciary does not return encrypted information directly but only references it by a pair of identifiers consisting of the identity of the system that owns the information and that system's accession number for the information. Basically then, the fiduciary is returning links demonstrating connections but does not return the actual encrypted information. This rule is optional. It delays the process but provides the peers with more control over their data.
3. The fiduciary maintains a so-called *checkpoint* wherein it tests the results against a series of access conditions that were established for the peer group. This approach can, for example, guard against the accidental distribution of large amounts of sensitive information due to a misunderstanding of the effects of the fiduciary's discovery process. The checkpoint is a refinement of the third-way method that forms the topic of a subsequent section.

These rules bring us to the core, essential fiduciary responsibility of the Blind Fiduciary. It only has access to encrypted data; therefore, it is not really holding the peer data in trust. However, the fiduciary has the responsibility to faithfully enforce the rules and procedures to which the peer group mutually agreed. These rules provide the peers with confidence that

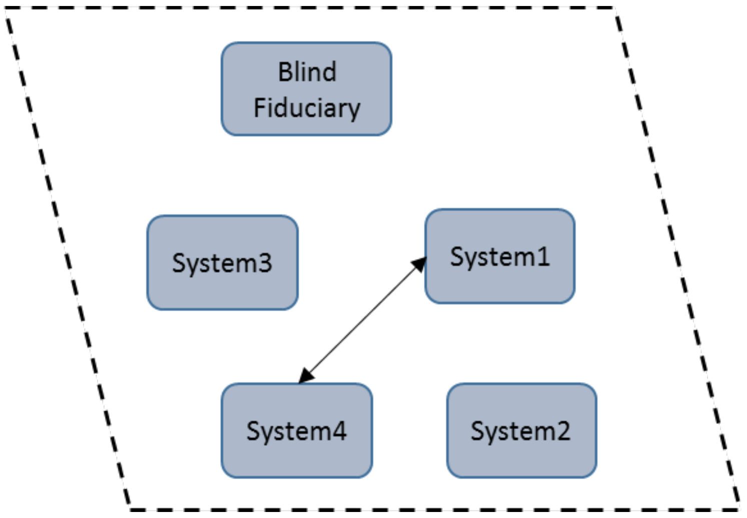


Figure 11. Use Case 4, Part 4, Variation 1. Systems 1 and 4 exchange information linked by a result from the blind fiduciary.

their interests will be respected during third-way operations. It is the fiduciary's responsibility to faithfully follow the rules.

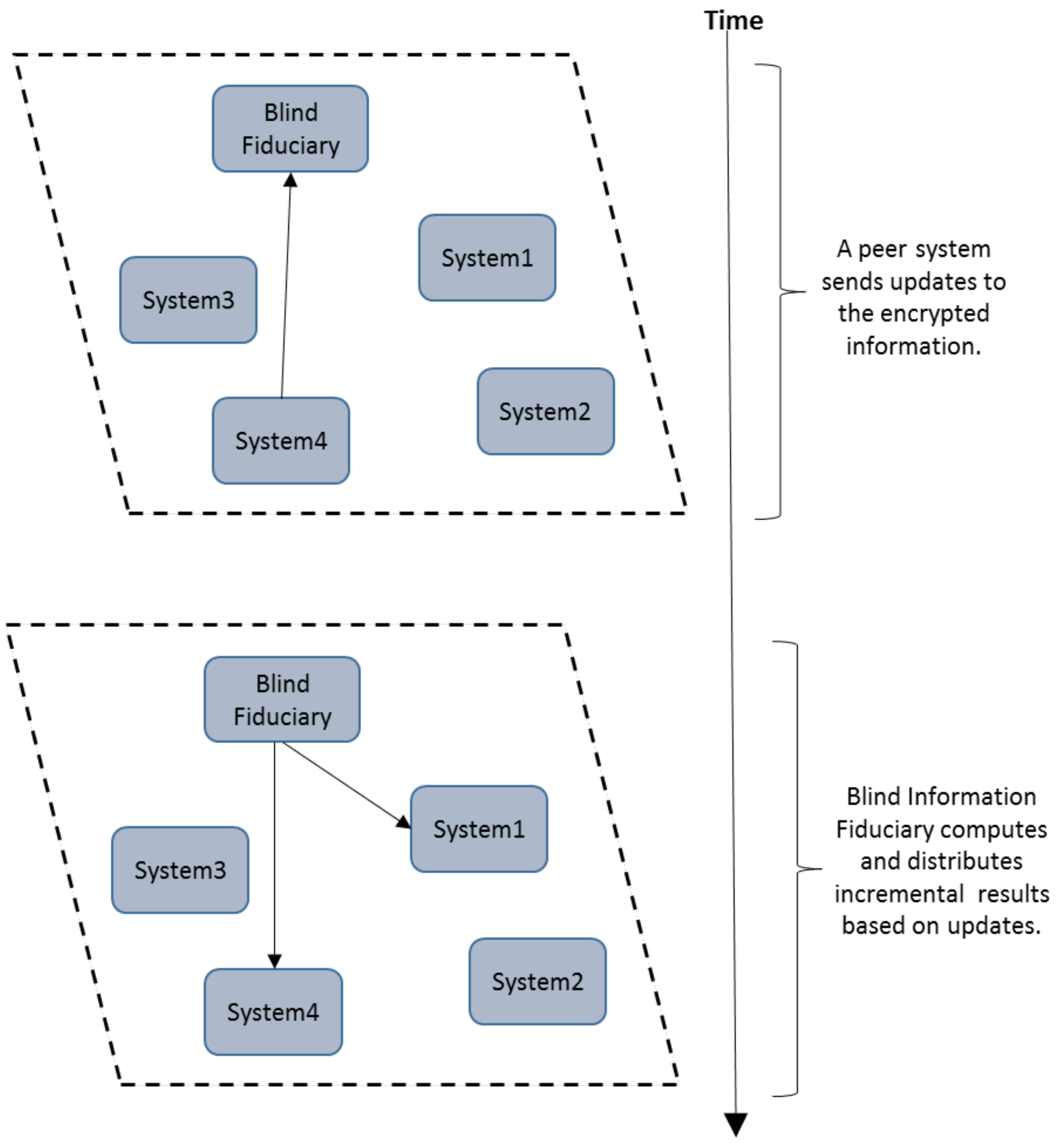

Time
Blind Fiduciary
System1
System3
System4
System2
A peer system sends updates to the encrypted information.
Blind Fiduciary
System1
System3
System4
System2
Blind Information Fiduciary computes and distributes incremental results based on updates.

Figure 12. Use Case 4, Part 5. The Blind Fiduciary accepts new information and produces new results until the end of the session.

To explain Use Case 4 further, we will assume that the fiduciary adopts the second rule: sending links. Then the owners of the actual information must exchange the actual information via a secure network connection as illustrated in Figure 11. Actually, the figure illustrates the simpler of two variations of this part of the Use Case 4. In this variation, the blind information fiduciary plays no role in this actual information exchange. The second variation is simple enough to describe: System1 and System4 use the

Blind Fiduciary (or really any other fiduciary) as a message agent. A message agent is an agent that accepts messages and delivers them to the addressee.

Variation 1 has the major advantage that it is direct; therefore, it is faster than variation 2. Moreover, variation 1 totally isolates the blind fiduciary from the information exchange. However, Variation 1 has the potential drawback that the transaction is refutable. By refutable we mean that an unfair peer can agree to exchange data, receive data from the other peer, and then renege on the promise to send data. The damaged party can make a note that the unfair peer is untrustworthy and bias itself against that peer in future negotiations. However, human nature demands an avenue for justice. Now at this point, demands for justice go outside our description of the system[8]. However, we can regard justice as a plug-in to be added later provided only that we require the message agent to keep an accurate, encrypted audit trail. That is possible with Variation 2.

As yet, Use Case 4 is not over. The session may last long enough that one or more peers has new information to contribute. Should that occur, the peer forwards an encrypted copy of the new information to the blind fiduciary which evaluates it for additional results. If the fiduciary finds results, it distributes them to the relevant peers. We illustrate this operation for Use Case 4 in Figure 12 where we assume that System 4 has new information that pairs with information from System 1. The fiduciary notifies both System 1 and System 4 of the linkage so that they may exchange actual data as illustrated earlier in Figure 11 on page 34.

At some point the session expires and Use Case 4 is finished. The duration of a session entails practical as well as security questions. As long as the session is open, the same session key will be used. Generally speaking, temporary keys should be just that, temporary. Therefore, the session should not last a very long time. On the other hand, each time the session starts, each system must adopt a new encryption key, encrypt all the data again, and send all of it to the fiduciary for this new session. Obviously that repetition entails an overhead, although the overhead is mitigated by the fact that the flow is easily handled by pipelined arrays of inexpensive hardware.

[8] *We return to the topic in the* Supplemental Topics*, specifically in the description of the* Adjudicator*.*

3. Applications

Note to the reader: This chapter has been assembled in haste. This draft should provide useful content but subsequent drafts will contain new material and may be organized differently.

Recognizing an Application

The following are specific applications in three general categories: Solidarity in Action, Vulnerability to Betrayal, and New Marketplaces. They can serve as examples of how to recognize an application.

Solidarity

In society, there are certain problems that remain unsolved because any solution would require a group action but the formation of a group is strongly discouraged. In this section, we discuss two such problems. Each problem has these contradictory attributes:

- Results require that independent parties work together.
- There are strong disincentives to working together.

Campus Rape

Society faces a number of difficult problems that might be solved with cooperative effort except there are disincentives to participating in the effort. One of these problems is relationship violence, even rape, on college and university campuses. Recent studies show the problem is widespread, it is growing and it is the victim rather than the perpetrator who has to fear the public exposure of the crime. Internet methods like social networking might, in principle, ameliorate the situation. In practice, they don't. The matter is too sensitive and private to trust to the open and easily exploited Internet. We need an improved Internet that protects secrets while sharing them when and where necessary.

Why the Campus Rape Problem Won't Go Away

If a woman is raped on campus, the crime will rarely be reported. Here are some valid reasons why the victim remains silent.

Reason 1: A significant, influential segment of the public blames the woman. For example consider the opinions in a recent syndicated column by George Will[9] . Mr. Will believes "Washington" (whoever that may be) wants to "make victimhood a coveted status that confers privileges" thereby inflicting irreparable harm on impressionable young men confused by "the ambiguities of the hookup culture, this cocktail of hormones, alcohol and the faux sophistication of today's prolonged adolescence of especially privileged young adults". If your father shares this judgmental attitude, would you want him to hear about what happened to you?

[9] Editorial by George Will, Washington Post, June 6, 2014: *Colleges become the Victims of Progressivism*. http://www.washingtonpost.com/opinions/george-will-college-become-the-victims-of-progressivism/2014/06/06/e90e73b4-eb50-11e3-9f5c-9075d5508f0a_story.html

Reason 2: What happened definitely does not enhance your status in any way; moreover, you will find a way to blame yourself. Better to keep quiet.

Reason 3: When you make an accusation, you face emotional and health stress. Your case is unlikely to come to trial. If it does, the judge will be biased against you. While the perpetrator goes free, you are left with hurtful memories. These opinions are buttressed by extensive statistics and surveys[10]. This is why victims keep silent.

You may be thinking: why not post the accusations on a web site anonymously (protecting the victim's privacy) and hope that the perpetrators can be shamed into better behavior or perhaps ostracized? Unfortunately, that online list would encourage another kind of bad behavior. The pettiest grievances could be avenged by posting an anonymous accusation against someone we want to hurt. The victim of the slander has no recourse. Hopefully, this fix will not be tried. For a serious accusation to be raised, the veil of anonymity must fall. It is sensible to maintain anonymity, however, until there is a reasonable, credible case that will stand up in court.

We know from the cases that have been reported that many transgressors are serial offenders. If multiple accusers come forward, a court must take the complaints seriously. However, at the moment, there is no way to maintain anonymity and bring together women who share a grievance. Before victims can find each other anonymously, we need to improve the Internet by adopting techniques of the Third Way.

Finding Solace with New Allies

No common database of sexual offense reports should be kept because a large, combined database is a tempting target for hackers. It would not be secure. The third-way solution is local and friendly. It is also indirect — working through an information fiduciary — to protect each user's privacy. Prospective users are encouraged to connect with a local group that they trust to keep a small, secure, local database of reports and act as their information fiduciary. As a member of a local group, you can report any offenses using a secure login to a computer or smartphone. For an illustration see Figure 13 on page 40. Your sensitive report is hidden – even from other group members. No information leaks to the "Big Data" cloud.

The information fiduciary supporting your group has several obligations. First, it operates a reporting service just for group members and maintains their privacy. Secondly, it works on behalf of its members to discover other offenses on campus that relate to the experiences of its members. The goal here is to find allies who have a common cause. When potential allies are found, each woman can decide whether to connect, discuss the mutual experience, and choose the next step.

---

[10] See, for example:
Editorial by US Senator Claire McCaskill, *Universities failing to protect rape victims*",
http://indy.st/1maBT9y
also:
the article by Tyler Kingkade, *Prosecutors Rarely Bring Charges In College Rape Cases*:
http://www.huffingtonpost.com/2014/06/17/college-rape-prosecutors-press-charges_n_5500432.html

Each group is completely isolated in terms of administration and data security. Nobody can login to a group and compare reports across campus. To find the connections between reports, we need a service that acts as a “blind matching information fiduciary”. Let’s explain that term and how the service works.

An information fiduciary is an agent that fulfills a set of obligations to its users and, when appropriate, to applicable regulations and laws. Each group has an information fiduciary that works for the members of the group and represents one group to other groups. Each group’s fiduciary agent can accept a report from a user. In addition, there is an independent “blind” agent that completes the solution. The agent is “blind” because it receives only encrypted summaries of the reports. It can’t actually read the reports and anyone who breaks into the third party agent cannot read the reports either.

The blind agent periodically checks in with all the agents for local groups. When the agent checks in, the group agents respond automatically by encrypting the reports with a one-time encryption key. Groups send the encrypted reports to the “blind matching agent” – but they never send the encryption key. Now the agent – operating “blind” because it can’t read the data – matches the reports and takes note of reports that refer to the same perpetrator. That is possible technically because a computer can match encrypted values, such as names and aliases, even when it can’t convert them to their original name or date or other meaningful fact.

The blind agent sends any matches to the groups that represent individuals. The matches arrive encrypted, but the group agent has the key to read the results. Thus, if you submitted a report that was matched with other reports, you will learn how the reports match. You will learn that others are in the same quandary for the same reason. You may decide to work with your fellow victim. If you decide to go ahead, you allow your name to be shared with fellow victims.

With the Internet working this way, victims can form alliances to go after the perpetrators. Punishing the perpetrators will reduce the incidence of crime. People working together can succeed.

An Illustration of a Shared Secrets Process

So far, I’ve given you an abstract answer to a concrete problem. To add an element of reality to the proposed third-way solution, let us consider how a woman might use the service. The following illustrations are screen snapshots taken from a Java-based application that implements the third-way solution. The implementation also serves as a demonstration and test for the Java software components discussed in the Tools chapter.

To create the illustration, we assumed that organizations available to students such as sororities and social clubs were willing to set up an information fiduciary that accepts and stores reports from the students securely. During the blind matching these fiduciaries that are distributed around campus will work together with the blind matching information fiduciary. The blind fiduciary might be operated by the college or perhaps by local law enforcement or even a national women’s rights advocacy group.

The users of this shared secrets process are women. Each has access to a secure smartphone app or to a login for a secure web server with a web page that acts as the front end for their local information fiduciary. The following figures illustrate an app registered to a fictitious woman, Eunice Yearby, who belongs to the Alpha Phi sorority.

In Figure 13, we show a snapshot of the user's application running for Eunice. A fellow named Tommy Kilcrease bothered her last year. He is the only person she has listed. Let's click on Tommy's entry and see what Eunice reported last year. This report is shown in Figure 14.

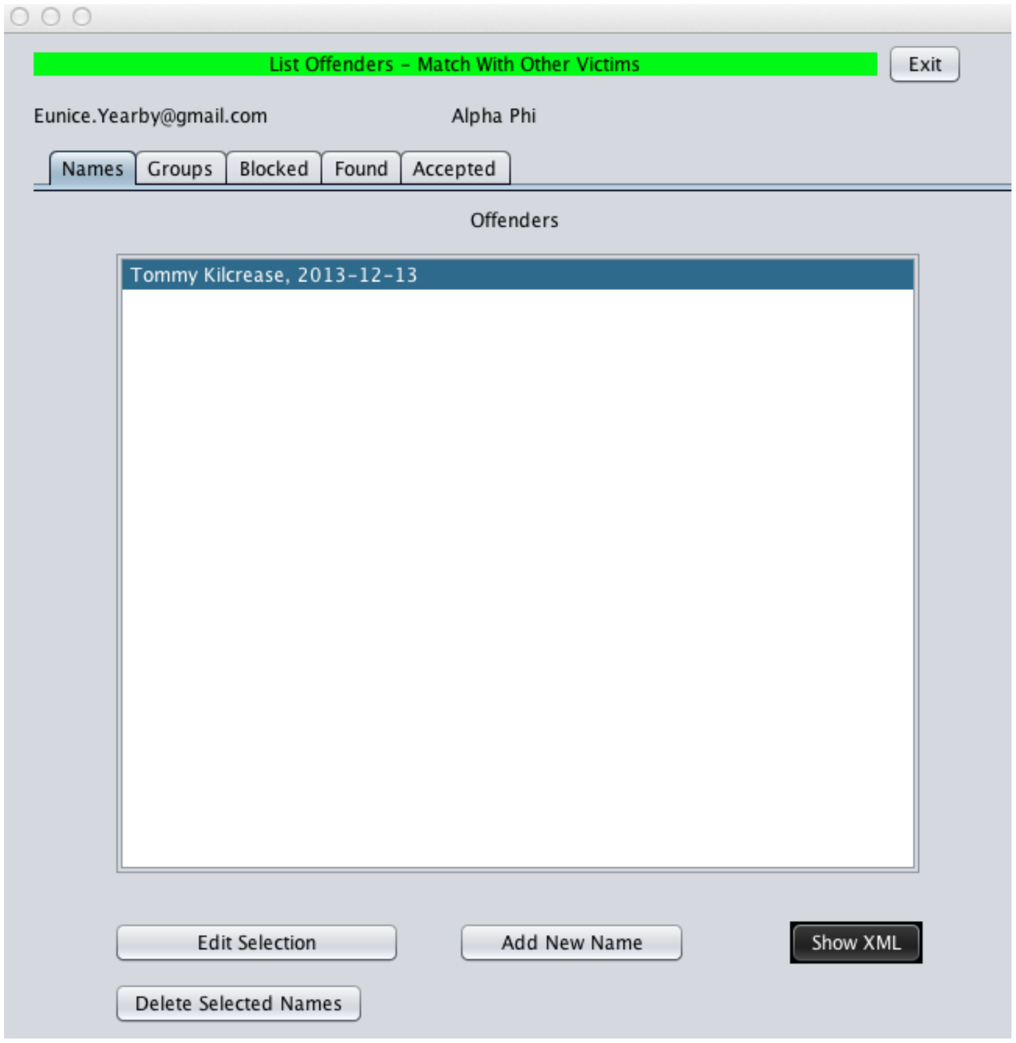
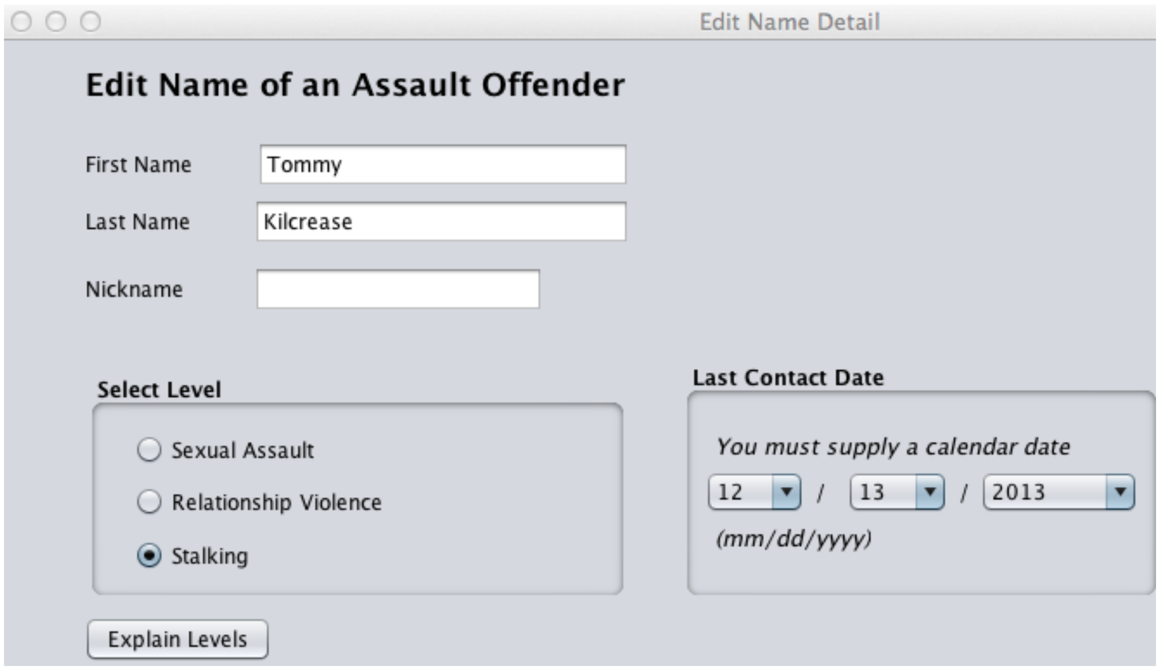


Figure 13. Illustration of an application for the user.

Our user, Eunice, has been watching for any other reports about Tommy by opening her application and clicking on the "Accepted" tab. As we see in Figure 15, this tab is currently empty. The Third Way protects Eunice's privacy by keeping the clear-text private and protected on her group's information fiduciary; that is, on the special web server and secure database operated by the Alpha Phi sorority. When records from different sororities are matched, the matching is done with encrypted records to protect privacy. We illustrate the record in question here in Figure 16. Note that the important fields are concealed by encryption. If a hacker gets the record, they are unable to use the names or dates.


Edit Name Detail

Edit Name of an Assault Offender

First Name	Tommy
Last Name	Kilcrease
Nickname	

Select Level

( ) Sexual Assault
( ) Relationship Violence
[✓ Stalking]

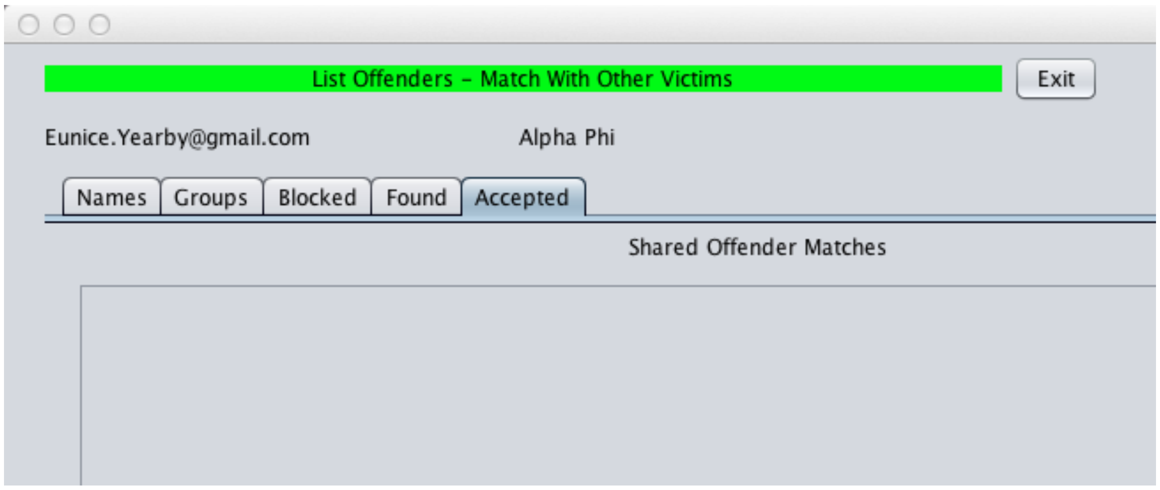
Last Contact Date

You must supply a calendar date

12 / 13 / 2013

(mm/dd/yyyy)

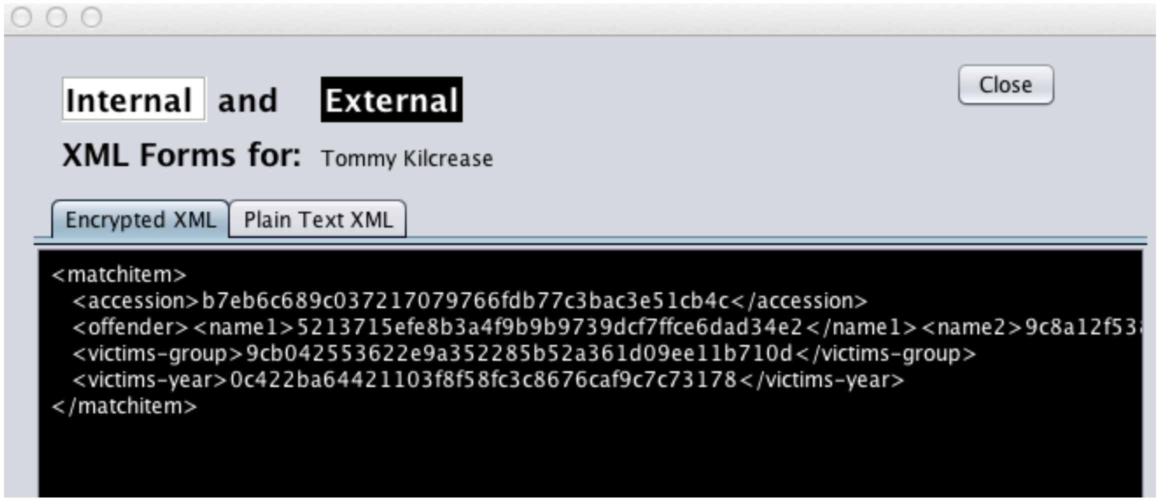
Figure 14. The entry form for offenders showing the details for Tommy Kilcrease. The level of the offense is coded with categories defined on the George Mason University web site. The "Explain" button would take you there.


List Offenders – Match With Other Victims

Eunice.Yearby@gmail.com Alpha Phi

Shared Offender Matches

Figure 15. The "Accepted" tab for the User's Application.



```
<matchitem>
  <accession>b7eb6c689c037217079766fdb77c3bac3e51cb4c</accession>
  <offender><name1>5213715efe8b3a4f9b9b9739dcf7ffce6dad34e2</name1><name2>9c8a12f53
  <victims-group>9cb042553622e9a352285b52a361d09ee11b710d</victims-group>
  <victims-year>0c422ba64421103f8f58fc3c8676caf9c7c73178</victims-year>
</matchitem>
```

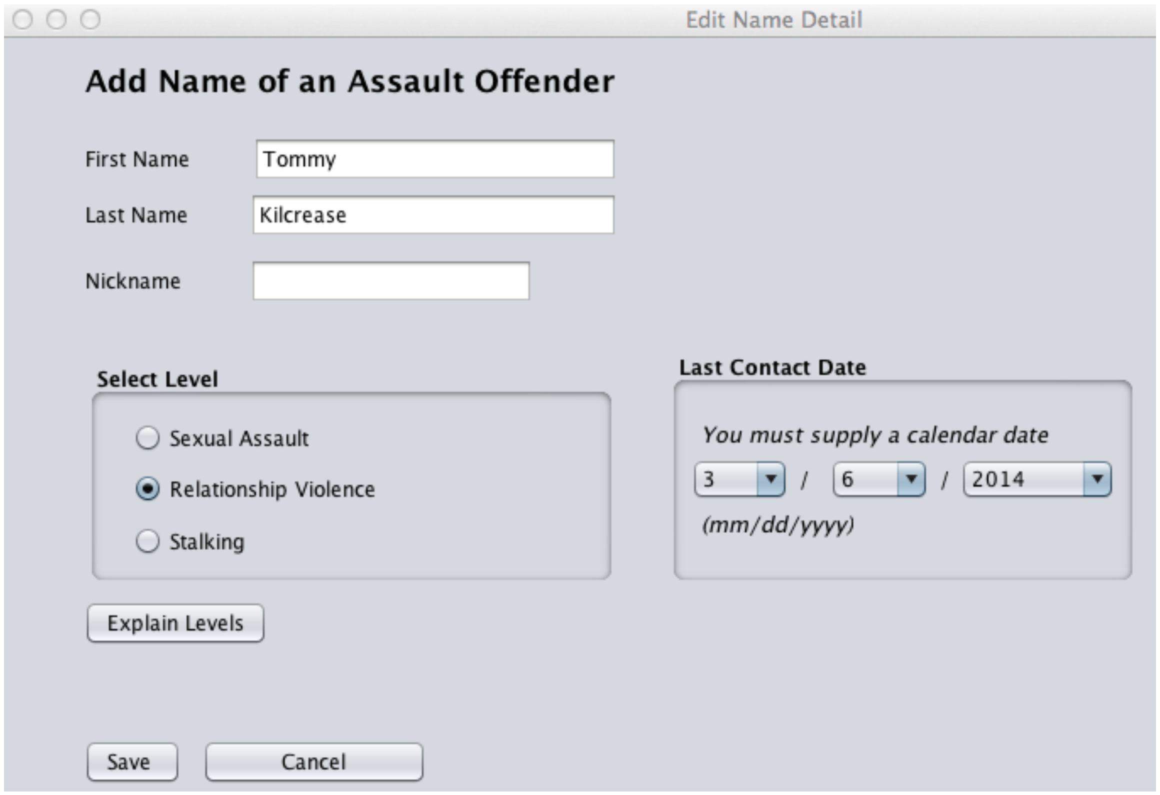
Figure 16. An encrypted incident record for Tommy Kilcrease held privately and securely for Eunice Yearby.

The reports that women file are held securely and locally. There is far less risk of loss when small amounts of sensitive information are held in many distributed compartments rather than a central database. Because compartmentalization prevents anyone from discovering related reports, a temporary encryption conceals the report so that it can be safely shared with an encrypted matching agent. In a production software system, the user would never see encrypted reports, but the demonstration software allows us, as users of a demonstration, to look at such encrypted data.

When Caroll Loden arrived on campus she joined the Kappa Alpha Theta sorority and, on the sisters' advice, registered for the system. She's been enjoying a safe, engaging learning experience so far and her screen for "Names" is blank. In fact, she has almost forgotten the system until she meets Tommy at a party.

Now Caroll has met Tommy, her attitude towards campus life is becoming one of fear and trepidation. In fact, fear is making it hard to learn. So she seeks support from other women with the same experience. The app lets her enter some details about her recent experience with Tommy. She clicks on the "Add New Name" button on the "Names" Tab as illustrated in Figure 13 on page 40. She can then enter details as shown in Figure 17. When Caroll saves this report, it appears in her report list; moreover, the encrypted version is shared with the campus-wide service.

You may recall from the discussion of Sessions on page 30 that the timing of any feedback from the campus-wide service will depend on timing specified by what variation of the session use case is implemented. For this service, it makes eminent sense to keep a session open so that a user adding a name will get immediate feedback. We have taken this option for the illustration. As a result, Caroll immediately sees that there is a match as shown in Figure 18.


Edit Name Detail

Add Name of an Assault Offender

First Name	Tommy
Last Name	Kilcrease
Nickname	

Select Level

Sexual Assault [✓ Relationship Violence] Stalking

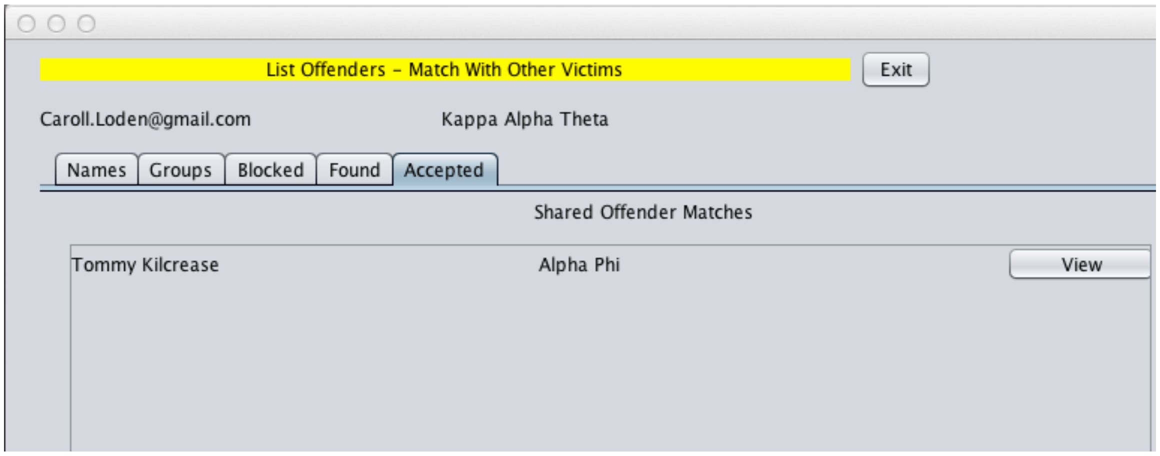
Last Contact Date

You must supply a calendar date

3 / 6 / 2014

(mm/dd/yyyy)

Figure 17. Adding a new report on an offender.

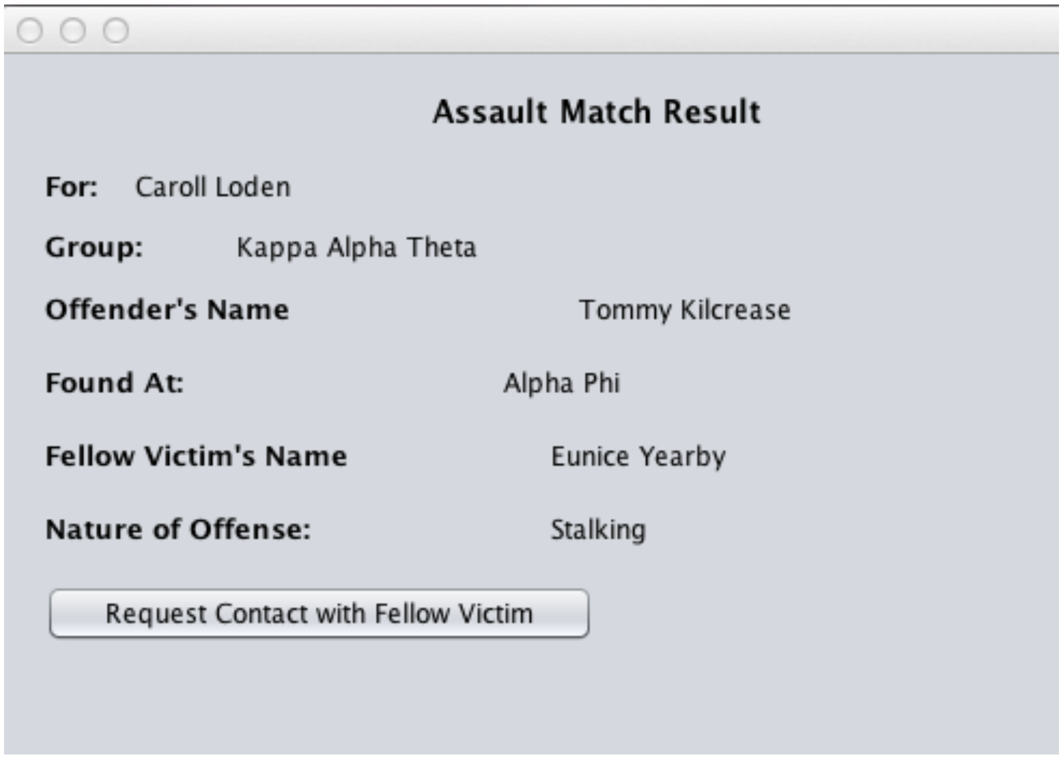


Caroll.Loden@gmail.com Kappa Alpha Theta

Shared Offender Matches

Tommy Kilcrease	Alpha Phi	View

Figure 18. The tab showing a match. The app notifies the user, Caroll Loden, in the Kappa Alpha Theta sorority that there is another victim's report by a user in the Alpha Phi sorority.



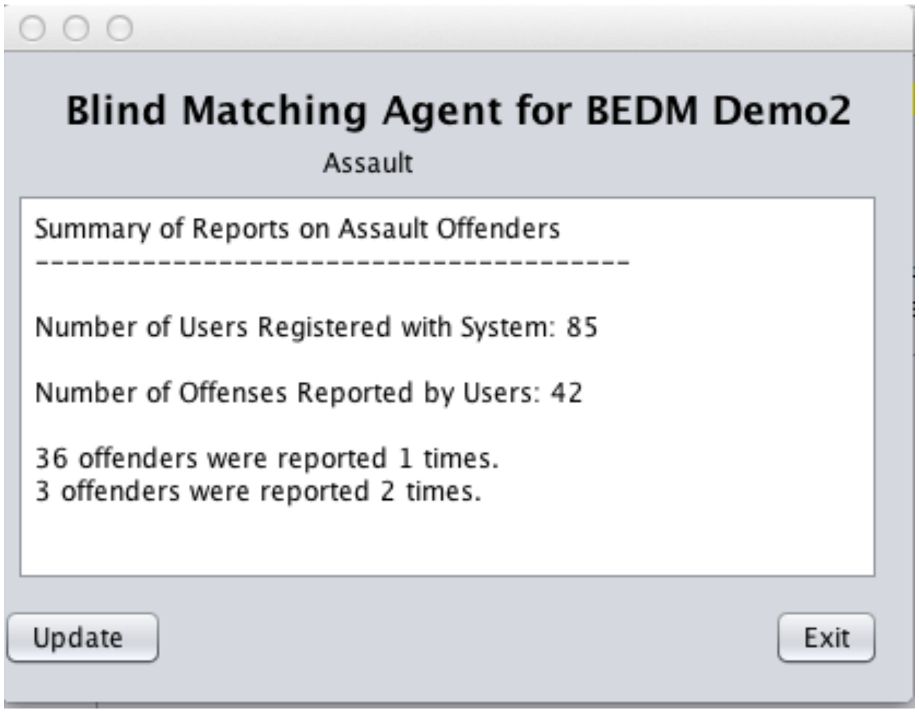
Assault Match Result

For:	Caroll Loden
Group:	Kappa Alpha Theta
Offender's Name	Tommy Kilcrease
Found At:	Alpha Phi
Fellow Victim's Name	Eunice Yearby
Nature of Offense:	Stalking

Figure 19. The match result was computed completely in encrypted form to protect the names but Caroll sees it after decryption in plain text. If Caroll decides to contact Eunice, she can send a secure e-mail message via the app by clicking the button.

Of course, on the other side of campus, Eunice Yearby will learn about Caroll Loden's bad experience the next time she opens the app. For Eunice, this will be news that she is not alone. In many real-world cases, we may expect to see three, four or even more matches on one offender's name. This system helps single out repeat offenders so they don't pollute the spirit of campus life.

Now over in the administration building, the Deans may be denying there is any problem. This system will not present them with any names, dates or facts unless the victims decide to come forward. On the other hand, the matching service can still count incidents even if it can't read the incident reports. If the administration asks for it, it can receive a report like the one we show in Figure 20 on the next page.



```
Summary of Reports on Assault Offenders
-------------------------------------

Number of Users Registered with System: 85

Number of Offenses Reported by Users: 42

36 offenders were reported 1 times.
3 offenders were reported 2 times.
```

Figure 20. Sample report from the campus-wide matching agent. Although the agent cannot read any details of the reported incident because they are encrypted, it can count. The counts of incidents may help engage the interest of campus administration.

These illustrations are derived from prototype software that you may also see demonstrated on YouTube[11].

[11] https://www.youtube.com/watch?v=LEcXOPuGzrs&feature=youtu.be

Callisto

It would be irresponsible to discuss the third-way approach to the campus rape problem without mentioning an alternative that partially achieves the same objectives and also goes beyond with an important problem-specific feature. This alternative goes under the name Callisto[12] and software has already been deployed at two college campuses in California. The software is available through an organization called “Sexual Health Initiatives”[13].

The Callisto approach follows the Third Way in that it is based on the selective sharing of private information only with parties that can contribute to a solution and only with the permission of the owner of the private information. The distinctive feature of Callisto is that victims make reports that are immediately encrypted, digitally signed, and registered with a notary system that can vouch for the authorship of the report and the date and time the report was filed. There is no way to post-date a report — a property that would be important to include in any matching system where there might be a motive to falsify or change a report.

Callisto diverges from the Third Way in that the protection is for victims only and not for the accused. In order to connect the encrypted, private reports of campus sexual violence, the Callisto system uses the names of the accused. This use of names in clear text opens the possibility that a student might be falsely accused by a conspiracy involving several women. The student thus accused would be presumed guilty on the basis of an association with multiple reports even if the accusers never unseal their reports. Thus, the system has a potential for abuse.

On the other hand, the importance of sealed, dated, and signed reports should not be ignored and that feature of Callisto should be included with any solution to the campus sexual violence problem.

Private-Public Partnership: Epidemiology

Motivation

The public and private sectors of the country each have information the other would find useful. One might expect a strong public-private partnership based tradition, shared history, and common interests. In practice, there is a working partnership for some issues while for other issue the two sides distrust each other and withhold cooperation. It is not our goal to explain or defend this observation. It should be clear there is some truth in it. Instead, we turn to an example where public-private cooperation might be improved: Epidemiology, the recognition and control of disease as it spreads in a population.

Example: Epidemiology

Suppose cases of a new disease pop up widely and randomly across the country. Unless public health officials act quickly to identify the individuals exposed to the new infectious agent, the contagion will spread rapidly.

For this discussion, we'll assume the US agency *Centers for Disease Control and Prevention* (CDC) takes the lead. The CDC must discover the chain of infection from the pattern of the reported cases. At the beginning of the investigation there is no obvious connection between cases.

[12] https://www.projectcallisto.org/
[13] http://www.sexualhealthinnovations.org/initiatives/

It is logical to suppose that the victims acquired the disease when they shared the same location at some time in the recent past. It would be simple for the CDC to pinpoint potential locations for disease transmission if a national database contained an inventory of all individuals in every location where people congregate. This hypothetical database would help the agency notify potential victims and isolate them if necessary.

On the other hand, consider where people customarily gather. They might attend the same professional conference, fly on the same airline, worship in the same church, or join the same political rally. In all likelihood, the professional society has attendance data and will share it. On the other hand, most Americans reject the idea of national databases for church attendance or political activity. Somehow, we must respect a general right of privacy while allowing selective exceptions for the common good or defense.

At the same time, the investigations of the public health officials cannot be completely transparent in their early stages. For every real, serious health threat, the officials will investigate numerous correlations or mistaken diagnoses that would cause unnecessary anxiety or even panic if made public. Thus both private and public sectors need to be careful with disclosure.

Discussion

We will use the airline industry as an example to explain the third-way approach because airlines already have good electronic data records. Consequently, there are fewer barriers to the implementation of this example.

At the start of the investigation, the CDC has only a list of victims. Obviously, the names of the patients and their medical histories are confidential. To investigate the outbreak, the CDC must identify airline flights and airports where the victims may have passed each other and acquired the disease. The location information is not in a central, national database because there are legal, organizational, and political obstacles to a monolithic database. On the other hand, there is ample justification to share data from confidential databases if it is relevant to a serious public health issue.

There is another important side to the privacy issue. The CDC's search must also remain a private matter until sufficient evidence accumulates to justify a public health warning. Most investigations will lead nowhere. If the public receives too many false alarms they will not pay attention to an important alarm.

The solution we offer to this problem is a procedure that finds the justification to share the information, elicits consent from the different authorities that control data, and then supervises the secure collection of just the facts relevant to this disease outbreak. In contrast, conventional encryption relies on access control lists to limit who may look at data but these conventional methods allow authorized parties to abuse their access privilege. That is why conventional methods trouble the citizens and they have good reason to fear widespread exposure or misuse of private data held by public authorities.

Summary

We have seen how an improvement in Internet facilities could allow people with similar concerns to find each other anonymously without revealing personal details on the web. Once they find each other, they can form an alliance and move forward as indicated by facts and circumstances. The service does require a significant improvement of Internet software. The effort, however, will be justified by the serious issues discussed here and by other applications that require both data privacy and selective, mutually-approved data sharing.

Vulnerability to Betrayal

The hardest argument to make for the Third Way is the argument for cooperation between independently governed agencies and entities that distrust each other. More often than not, there are reasonable, valid expectations that one party will betray another somehow, sometime. In the past, the entities took the easy road, hunkered down, and dealt with the other party at arm's length giving nothing away. The case for taking the Third Way rather than this easy road rests on two arguments.

First, both you and your unreliable neighbor probably face some bigger more menacing threat that is a danger to both of you. You are stronger together than at odds with each other. Or, to put it another way, the enemy of my enemy is my friend.

Second, you should always assign weights to threats. Sure, your neighbor and potential ally is a risk, but in many cases, the people who work for you represent greater risks. If you really worry about risk to sensitive data, then you should be adopting the Third Way for internal use. Once the Third Way's infrastructure is deployed, cooperation with an outside entity is a simple extension and no less secure than your internal operation.

Nevertheless, fear persists (justified or not) and fear inhibits cooperation with your neighbor. We will leave it to the psychologists to deal with the human aspects of organizational behavior and carry on here with the technical side.

The following illustration is one that I prepared for a government agency. It is noticeably context free. There was no other way. When an agency's work is highly classified they cannot share even false information that betrays the kind of real information that they handle. I hope, however, that the reader can imagine how the illustration demonstrates that sharing is possible while maintaining security.

A Context Free Illustration

The software for the Third Way has been tested in a variety of ways; however, some potential users have mentioned a desire to see the software operating in an exceptionally simple, context-free demonstration that uses generic textual data from the Internet and focuses all the attention on the unique aspects of the Third Way. The technology used in a simple demonstration can be easily evaluated because there is no need to consider a social, economic, or political context. When we say that the demonstration is context-free we mean that:

- there is no application domain
- there is no representation of geographic, security, or political boundaries in the enterprise data system
- there is no commitment to a specific enterprise architecture
- there are no special software technologies other than the essential and distinctive procedures provided by the third-way software

Here, we discuss the data, the matching method and the operation of the demonstration.

General Plan for Demonstration Zero

The demonstration models a distributed application running on multiple sites and accommodating host systems that play different roles. To keep it simple, we have just two host systems that are independent peers that own sensitive data. We call these the Blue Team and the Green Team. To act as the blind information fiduciary, we model a third host that serves that role. For this section, we will call that host the Blind Agent Negotiator (BAN) because its function is to negotiate a transfer of sensitive data between the peers.

Finally, we model a host that serves a supervisory role in the demonstration. The supervisor starts and stops the demonstration and opens various window displays to show the internal operation. In a real system, administration of the operation is a more complex job.

The presumed backstory for the demonstration is that the Blue Team and Green Team work separately, they do not trust each other, and they have not been cooperating. Each possesses documents that may relate to documents held by the other Team. Both teams would benefit if they exchanged related documents, but both teams see a security or mission risk if they freely exchange all the documents. Thus, they have a motive to employ the services of the BAN to match encrypted documents and identify those that are related.

In this demonstration, the BAN does not see the actual document text, even in encrypted form. Instead, the BAN must identify related documents by examining encrypted word counts derived from the documents. After a match of related documents has been found, the actual exchange of secure documents must be handled through other channels. It would have been easy for the BAN to receive encrypted documents and provide the matching ones to the Blue and Green Team. We did not configure this demonstration for that option because we wished to follow the most conservative, most secure sharing policy.

Demonstration Data

For this demonstration we will match related text documents using only encrypted information. Our method for matching is a simple formula based on word counts in the documents. Documents that use similar, distinctive words are regarded as related. Of course, an effort is made to avoid an influence from common words like "a" or "the", which appear in all English documents.

We obtained our documents from the public domain of old documents that are not subject to any copyright restriction and downloaded them over the Internet from the site authorama.com. Each document is an essay written by the same author, Francis Bacon, in the 17th century. There are 59 such essays[v] and we have handled them in a way that divides the collected essays into 59 incomplete essays for the Green Team and a different set of 59 incomplete essays for the Blue Team.

The story line for the demonstration is that the Blue Team and the Green Team receive fragmentary information about a subject and the only way either Team can put together all the information and read the full essay is by finding the missing related information in the other Team's possession.

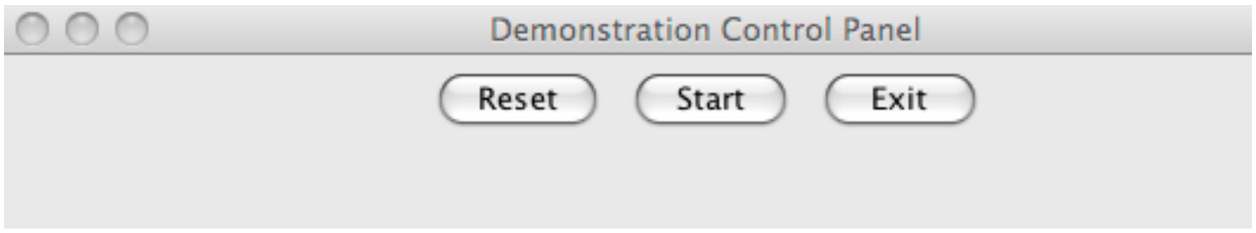
To create this initial distribution of secret data, we took each essay and divided the text into blocks of 50 words. Even numbered blocks went into one new, abridged essay and the odd numbers went into a separate abridged essay. Each team received half the word blocks in the original essay.

While we preprocessed each essay into two distinct abridged essays, we also formatted the documents for XML data exchange. Formatting is simple. There is a body containing the text and a title. In addition, we calculated word counts for the text body and placed the word counts in the XML document. At the end of the preprocessing, each Team has XML information for 59 essays and the associated word counts. That information is the starting point for the demonstration.

Demonstration Roles and Display Screens

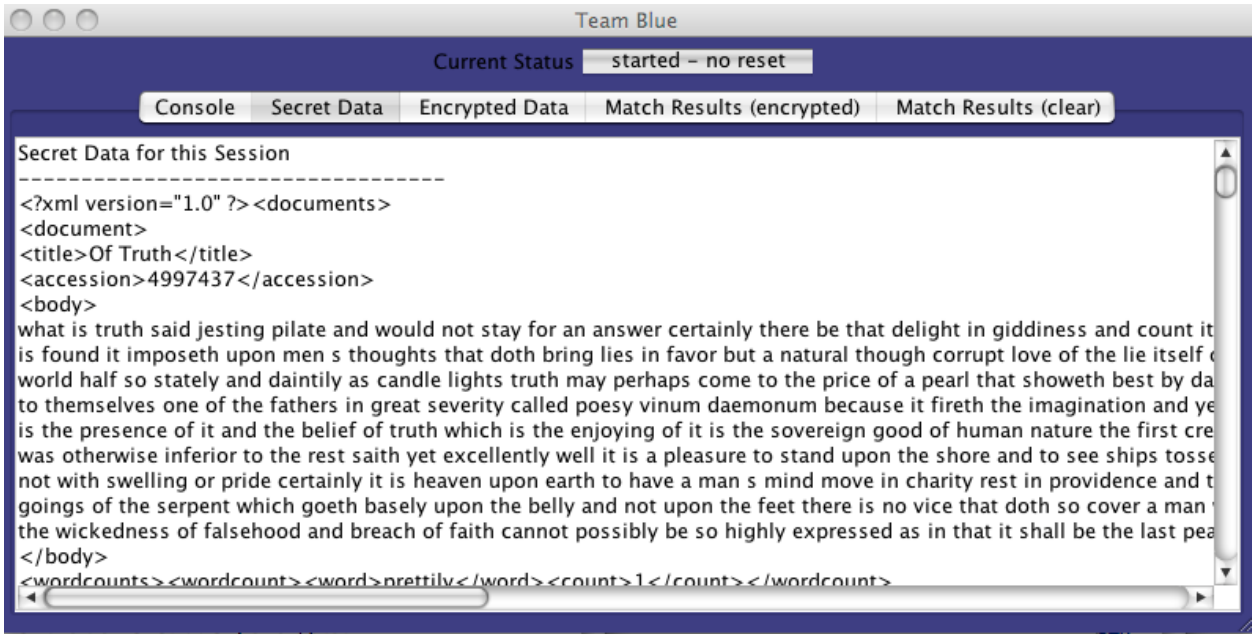
1. Operations Manager

The whole operation is supervised by a computer user who runs the demonstration. The software provides a control panel that contains buttons to initiate three operations:



2. Blue and Green Teams

The two teams manage the data that they have acquired and protect from outside access. For the demonstration, the data holdings are static. In any real-world operation, however, the holdings would grow as documents come in from the field. For each team, there is an independent software object that runs in response to encrypted messages from the other agents in the demonstration. This software object is visible on the demonstration screen as a window that shows the status, the processing activity, and the data documents for the Team. The following picture shows the window for the Blue Team:



```
Secret Data for this Session
----------------------------------
<?xml version="1.0" ?><documents>
<document>
<title>Of Truth</title>
<accession>4997437</accession>
<body>
what is truth said jesting pilate and would not stay for an answer certainly there be that delight in giddiness and count it
is found it imposeth upon men s thoughts that doth bring lies in favor but a natural though corrupt love of the lie itself
world half so stately and daintily as candle lights truth may perhaps come to the price of a pearl that showeth best by da
to themselves one of the fathers in great severity called poesy vinum daemonum because it fireth the imagination and ye
is the presence of it and the belief of truth which is the enjoying of it is the sovereign good of human nature the first cre
was otherwise inferior to the rest saith yet excellently well it is a pleasure to stand upon the shore and to see ships tosse
not with swelling or pride certainly it is heaven upon earth to have a man s mind move in charity rest in providence and t
goings of the serpent which goeth basely upon the belly and not upon the feet there is no vice that doth so cover a man
the wickedness of falsehood and breach of faith cannot possibly be so highly expressed as in that it shall be the last pea
</body>
<wordcounts><wordcount><word>prettily</word><count>1</count></wordcount>
```

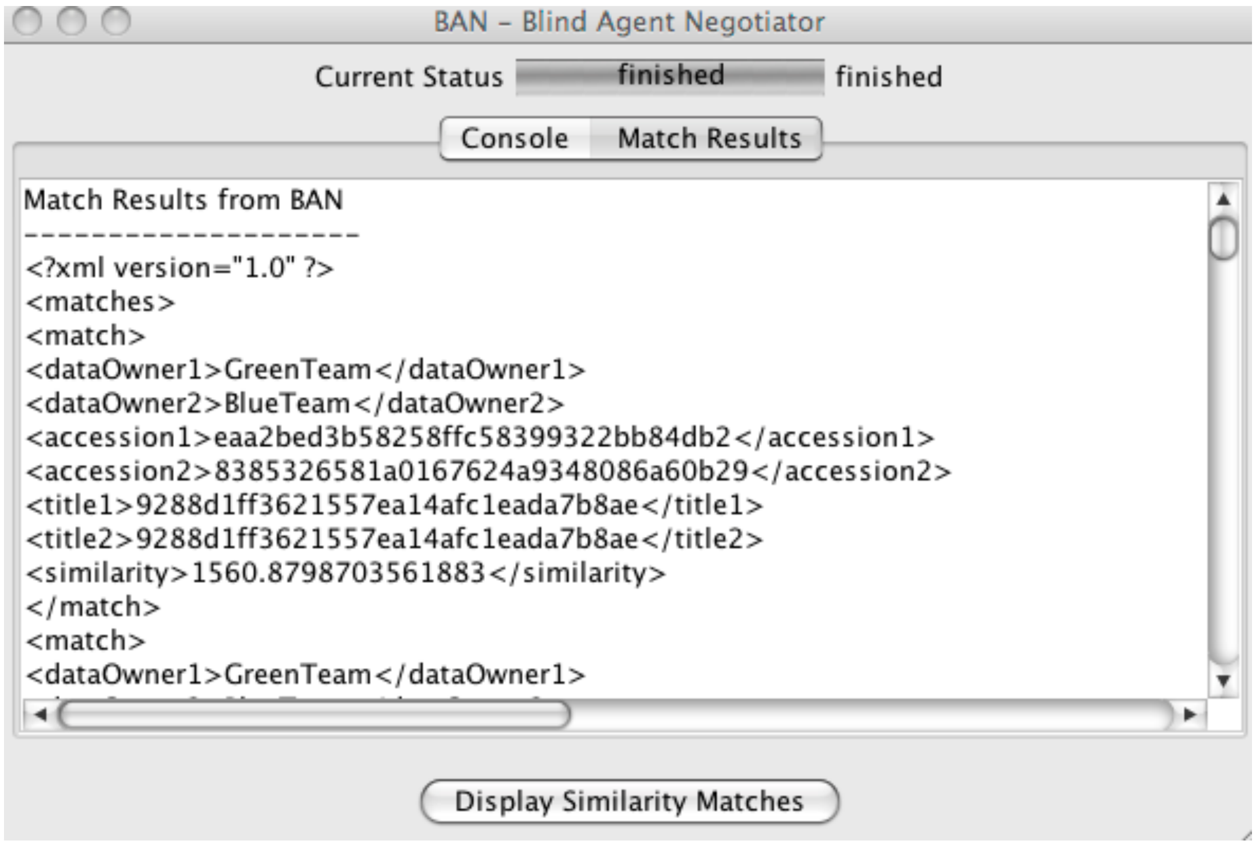
The window for the Green Team is identical except for the obvious changes to the name and color of the panel. Each of these Team windows has a tabbed text area that can show: *Console* (progress messages from the software), *Secret Data* (the secure hidden data that the Team has acquired), *Encrypted Data* (the same secure data after encryption by the session key), *Match Results* (*encrypted*) (the match results from the BAN protected by the session key), and finally *Match Results* (*clear*) (the match results from the BAN in clear text.

The line above the tab bar is the status and progress line. It will always show the current status of the team during the matching session. While the team's software is actually processing data, the area changes to a progress bar.

3. Blind Agent Negotiator - BAN

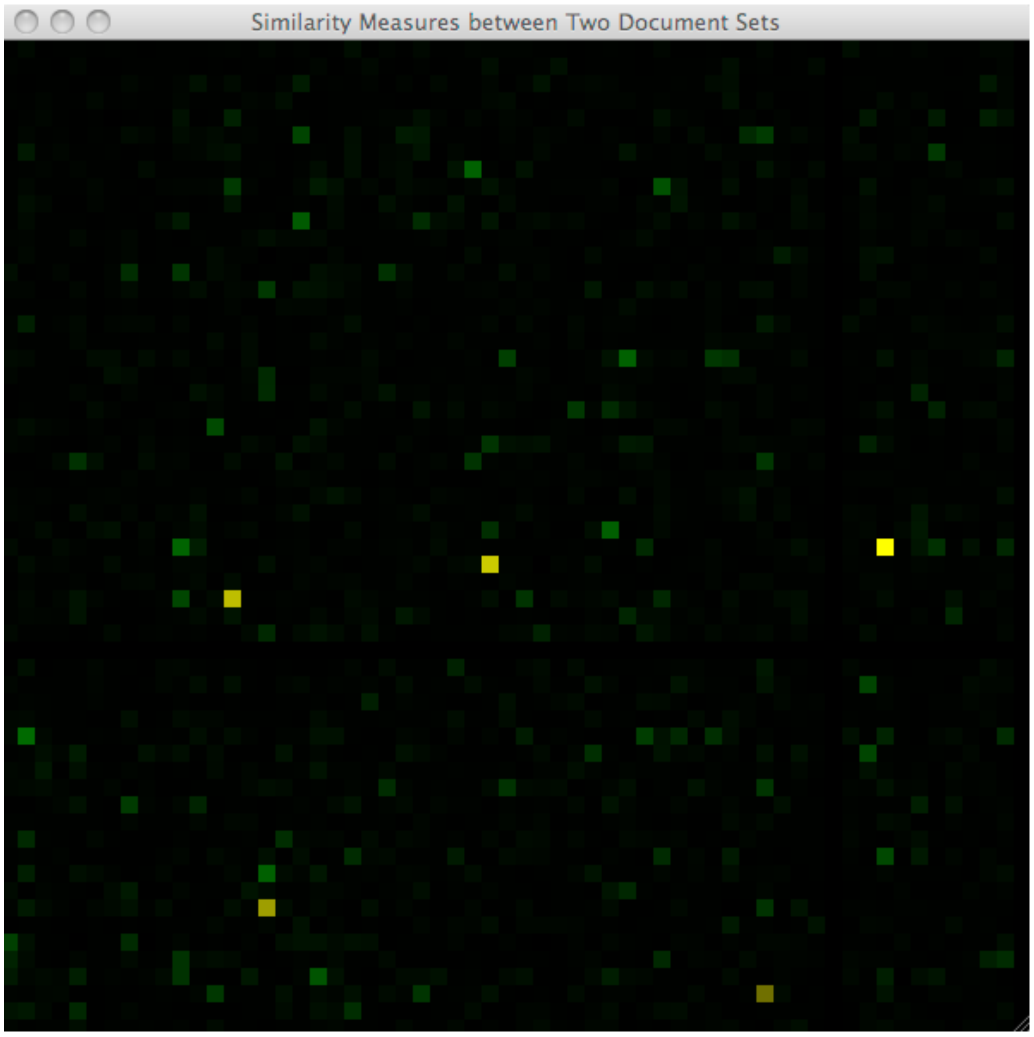
The BAN plays the role of the middleman in the Third Way. It is implemented by a software object that runs in response to encrypted event messages and it is visible in the demonstration through its software window. Below, we show a snapshot of the BAN's window taken after the conclusion of the matching process. The text area shows part of the match results to illustrate that the result of the match is written in well-formed XML with encrypted fields protected by the session key.

When the match is finished, a new button appears at the bottom of the window: *Display Similarity Matches*. The button is not visible at other times because the information is transient and it is not available before completion or after a “Reset” or “Exit”. The full output of the matching operation is held temporarily as part of the demonstration.



```
Match Results from BAN
--------------------
<?xml version="1.0" ?>
<matches>
<match>
<dataOwner1>GreenTeam</dataOwner1>
<dataOwner2>BlueTeam</dataOwner2>
<accession1>eaa2bed3b58258ffc58399322bb84db2</accession1>
<accession2>8385326581a0167624a9348086a60b29</accession2>
<title1>9288d1ff3621557ea14afc1eada7b8ae</title1>
<title2>9288d1ff3621557ea14afc1eada7b8ae</title2>
<similarity>1560.8798703561883</similarity>
</match>
<match>
<dataOwner1>GreenTeam</dataOwner1>
```

During the matching, the BAN compares each encrypted document from the Green Team with each encrypted document from the Blue Team. The BAN calculates a similarity index for each pair of documents. The BAN selects the 5 most similar documents and notifies the Blue and Green team about those pairs of similar documents. Meanwhile, the BAN is still holding all the similarity comparisons between documents. When you push the “Display Similarity Matches” on the BAN window, the software uses the temporary comparison data to display the full results. The following window illustrates the display:



The visual display shows a matrix of values where the documents from one team are arranged along the horizontal axis and the documents from the other team are arranged along the vertical. Each color square in the picture represents one comparison. The color depends on the similarity of the two documents. The brighter the green, the stronger the similarity. The computer marks the five most similar documents by shifting the color from green to yellow.

It is worth repeating that the visual display of the matrix of values shown above is an illustration produced to help explain the matching process. The teams receive notification when documents match and that notification contains only a pair of accession numbers. One member of the pair is an accession number of one of the recipient's documents and the other accession number points to a document in the possession of someone else.

New Marketplaces

During the 1950's in the early days of television, quiz shows were very popular. Two popular shows, "What's My Line" and "I've Got a Secret", featured a guest with a secret, who answered questions from panelists. The panelists would try to guess the secret based on the responses to their questions. These old entertainment games have surprising similarities to many business processes. Competition forces each player to understand the other players. Each player will try to learn what it can by asking questions about prices, contracts, and other available information. Perhaps you have a web site advertising your prices. The prices give away information about your costs because baked into your price are cost factors like:

- Your wholesaler's price to you as the merchant for your current inventory.
- Costs of maintaining the inventory.
- Anticipated cost to replenish inventory.
- General corporate costs such as: Interest payments on any debts, payroll, rents, etc.
- Desired profit margin.

Your competitors would like to know your cost factors so that they can best set their prices to compete with you and optimize their profits. If the costs are completely open to public inspection, then nothing prevents your competitors from using public information to optimize their price strategy.

We could have extended the list of cost factors to include the following:

- How much inventory is left?
- Can inventory be replenished quickly?
- Do you plan to continue the product or discontinue it when the inventory is sold?

These factors lead to other considerations. For example, suppose you establish a low price and the product sells so well that your inventory is exhausted. Buyers are turned away or their requests are placed on backorder. You know you may lose customers to a competitor if they are placed on backorder for too long or too often. This concern raises an issue that is critical to actual producers rather than sellers: what is the size of the order?

If you actually need to produce the item sold, then you need to worry about:

- Cost of materials.
- Availability of production equipment.
- Cost of additional production equipment
- Labor cost – both regular and overtime hours.
- Setup time.
- Delivery schedule.

All these factors are competitive information that is not something you can freely share. As a result, orders to producers — unlike sales to the end-consumer — are usually done by a paper bidding process using manual rather online methods. The paper bidding and careful negotiation by people can lead to fair contract terms that meet the needs of both the buyer and producer at an agreeable price. The ability of negotiation to optimize the results is a benefit that we can hope to extend into the sphere of automated activities by following the Third Way.

There is one other reason that much of the business process is not automated: reputation. If a seller has a reliable buyer who always pays an invoice on time, that buyer gets a preferred price. Another buyer who pays late or pays less than the invoiced amount won't get the same deal. If a company's reputation is poor, the prices it sees are marked up to reflect the risk of doing business with an unreliable partner.

During a negotiation it is rude and unproductive to discuss reputation openly particularly when the implication is clearly that one client is getting better terms than another. In one-on-one negotiations however, the matter is private. The two parties agree on their bilateral terms and the terms offered to others do not interfere with the negotiation. That scenario naturally lends itself to automation by the Third Way. The Third Way arranges bilateral agreements while keeping the history of other agreements secret.

Example: Keeping Your Cost Factors Proprietary

Success in business depends to a large degree on knowing what will happen in the future. For that reason, businesses select the most intelligent, experienced leaders who will make the best educated decisions. Or, perhaps they select someone who made lucky predictions in the past and might have luck in the future.

Below the top level however, down in the nitty-gritty detailed part of business, the situation changes. The business must deal with cost factors that move with the season, with events like floods and droughts, with new technology, with regulatory changes, etc. For these more detailed decision factors, it makes good sense to model the future using past and present data. Quantitative analysts can predict the near term surprisingly well using statistical methods and parameterization of cost-factor networks. But their judgement doesn't come cheap and their results are yours only so long as you can keep the results to yourself. Your understanding of cost factors and models becomes your proprietary secret, your competitive edge. In short, knowledge is a valuable, proprietary business asset.

If competitors can learn enough about your cost model, they may find a way to exploit that knowledge about you to their advantage. Thus, you want to keep that information from them. Competitors can watch the moves you make in the marketplace and gain an understanding of how your internal, proprietary model works. That is something you don't want.

That is why a responsible buyer will establish a bidding process that claims to keep your bid private. The details of the deal negotiated between buyer and seller are ideally a private matter and bid information is used only for the evaluation in a particular procurement. If the bid terms become public, they expose proprietary competitive information.

So can we make an online system that results in fair contracts and preserves competition-sensitive information? Yes: we illustrate the answer here with a fictitious market for stock swaps.

An Illustration with Stock Swaps

You have never heard of a market for stock swaps; none exists. But you probably don't know the market for credit default swaps either, although it is real. You can trade almost anything if the market seems fair. Whether or not people trust a market often depends on the "market maker" — someone or some organization that brings order and reasonable expectations to the transactions in the market. In our illustration, the market maker for stock swaps operates a blind information fiduciary that finds agreeable contract terms using only encrypted information about the terms of requests and offers.

The stock swaps in this example work analogously to currency swaps. In a stock swap, one party asks to exchange a certain number of shares in one listed stock for a cash-equivalent number of shares in another listed stock. This party, called the “buyer” offers cash as an incentive for another party, called the “seller” to enter into a stock swap agreement. A stock swap agreement has a term of a defined number of months. At the end of the term, the swap is reversed thereby closing the transaction. The second party keeps the cash paid by the first party when the swap was initiated.

To be clear, at the start of the swap the participants exchange a certain number of shares in two different companies. At the end of the swap, they return exactly the same number of the same shares. The actual profits and losses depend on the market changes during the term of the swap; that is the value of each share of stock will usually change during the swap. Each party bears the risks of the position it assumes.

Obviously, success in this market is a measure of each player’s ability to predict detailed changes in the stock. The participants in the market believe they can win and make a profit because they are particularly adept at prediction and/or have a better predictive model. There may be other reasons to participate. You may have an exposure to changes in one stock but can’t change your position in that stock easily. Then a transaction in the stock swap market can help insulate you from changes in that stock.

The Request to Buy a Swap

In any real market, many people are looking to buy and many to sell. In this illustration, we’ve simplified matters. First of all, we swap only the stocks that are included in the Dow Jones Industrial Average, DJIA and there are only a few of those. Second we assume only 4 players, A, B, C and D, who are offering to sell swaps. Only Player A is looking to buy a swap at this time and there is only one swap that Player A wants — a swap of the Disney stock that A owns for IBM stock that some other player owns. We will explore in detail only the search for an acceptable contract on that one stock swap — the swap of Disney for IBM initiated by A and completed by a contract with B, C, or D.

Every player has some flexibility on price. However, it is bad negotiating practice to state up front just what range you can accept. Furthermore, players don’t really want other companies to know too much about their business. Ideally, no one should know that Player A wants to swap Disney except the player that agrees to the swap. Thus, the players keep their negotiating parameters under strict lock and key. In the software that simulates this illustration, we provide a “god’s eye view” that allows the spectators to look at all the players' hands without any security impediment. In a real application, the god’s eye feature would be blocked by the security provisions.

So let’s look at what Player A wants:

```
ask:
  have: DIS
  want: IBM
  shares: 10000
  cash_per_share: [ 0.0000, 3.4100]
  interval: 12
  ref_id: 12345
end
```

This is a formal request by Player A to buy a stock swap that exchanges Disney stock for IBM stock for a period of 12 months. The buyer offers 10000 shares of Disney for a certain number of IBM shares to be determined by the relative prices of Disney and IBM stock at the opening of the stock exchange. In addition, the buyer offers to pay as much as $3.41 per share ($30,410) to the seller who accepts the Disney stock in exchange for IBM. Obviously, the buyer would prefer to pay less, ideally nothing! Player A wouldn’t want to start by showing a willingness to pay that much up front. However, in this case, the negotiation is carried out by exchanging encrypted documents with a blind information fiduciary acting as a market maker. This blind fiduciary cannot read the documents. Consequently, Player A can write down their company’s negotiation position including the flexibility on price and still be confident that neither the market maker nor the opposite players knows the exact terms of the position.

The Agreement

Let’s jump ahead and see what the eventual negotiated agreement looks like. In this example, the blind agent operated by the market maker finds that there is a match between Player A, as seller, and Player B, as buyer. The mutually acceptable terms of the swap agreement are:

```
agreement:
  shares: 1.89737113675e+22
  swap: 236368f3e9e96745731d5d7a4c01d923c931e12688a685bae4f0c8ebf47d31d2
  for: 46914567b8c8b041cf54170128534b77b650684e46f31e541884d1a63d3e40cd
  plus: 8.89199078881e+17
  interval: c1cf3b39c755de7cb8fc52d3f03554d3851efed0ca8fc8930e3fdecdbdd97f06
  ask_ref_id: 12345
  offer_ref_id: 177379
end
```

O.k., it doesn’t look like much because it is encrypted. That is the idea. The market maker finds the agreement but the market maker has no idea what the agreement contains. That is what the Third Way is all about: players finding and executing a transaction without putting sensitive information at risk.

The market maker sends the encrypted agreement to Player A and Player B, both of whom know how to read it. In clear text, it says:

```
agreement:
  shares: 10000.0
  swap: DIS
  for: IBM
  plus: 1.705
  interval: 12
  ask_ref_id: 12345
  offer_ref_id: 177379
end
```

The agreement is that Player A swaps 10000 shares of Disney plus $17,050 dollars for a number of IBM shares that have equal value on the date of the swap. Player B provides the IBM shares, receives the Disney stock and keeps the $17,050. Player's A and B agree to simply exchange the stock at the end of 12 months without regard to any changes in the price of IBM and Disney over the interval.

Matching Requests and Offers

The role of the market maker is to match pairs of players. One of the pair requests a swap and thereby initiates the negotiation process while the other member of the pair completes a swap by offering terms that are acceptable to both members of the pair. We have seen a typical "ask" thrown on the market by Player A. Now let us look at a typical "offer"; in fact, we will look at the offer from Player B that matches the request:

```
offer:
  accept: DIS
  for: IBM
  cash_per_share: [ 0.000,  77.6159]
  number: [7507, 60060]
  interval: 12
  ref_id: 177379
end
```

A serious player like Player B will not make an offer based on intuition or general policy. A serious player has a model that predicts the direction of the stock prices, takes into account the existing positions the player has assumed to avoid being overextended, and folds in the player's risk tolerance and profit expectations. The model computation will be complex but it boils down to a few numbers in the offer. The c*ash_per_share* range captures the profit aspect. In this example, the bottom of the range is zero which means that Player B expects to make a profit on the swap due to future changes in stock prices. However, Player B will also put in a high value of *cash_per_share* to capture more profit if Player A is willing to complete the transaction at a higher *cash_per_share*.

Let's look now at the *number* range in Player B's offer. Roughly speaking, the *number* range in the offer captures the model's calculation of risk tolerance. No player should offer a transaction so large that it might cause irreparable harm to the player if the player loses money on the swap.

The market maker sees the encrypted version of the "ask" and the "offer" and matches them. Let us instead examine the match in clear text. The "ask" is for 10,000 shares which lies in the range of the offer, [7507, 60060]. The "ask" is willing to pay in the range of [0, 3.140] per share and of course the offeror would prefer a lower number. The player making the "offer" would like to get *cash_per_share* between zero and 77.6 and would prefer a higher number. To match these the "ask" and "offer" the market maker checks that the number of shares is acceptable to both players and that the *cash_per_share* lies somewhere near the middle of the expectations[vi] of the "ask" and the "offer". When it finds the match, it proposes the agreement to the two parties who can then decide whether to complete the transaction or not.

More info

This section on stock swaps is based on older material[14]. It will be completely rewritten in future drafts of this book. What you see here is derived from on-line documentation. If any reader cares to explore this illustration in more depth, the on-line documentation is more extensive. A good starting point for reading may be:

http://www.wwnsoftware.com/demo1/BEDM_Detail/Broker_Preparation.html

Summary of Advantages

Here is a quick list of the advantages shown by the third-way negotiation for stock swaps:

- **Strict privacy:** Players A and B reached an agreement on a stock swap without conveying any information to the other players, C and D. The broker knows an agreement was found between A and B but the broker doesn't know what it says.
- **Automatic price negotiation**: The concealed price ranges of players A and B were successfully reconciled by the broker to reach a mutually acceptable price without letting A and B know how flexible the other side was on price.
- **Protection for the stock price mode**l: If anyone could look at all the requests and offers that the players are making, they could deduce a player's predictions for the stock market. However, the requests and offers are encrypted so each player's price model is safe.
- **No built in fairness**: Older on-line systems like Amazon.com are fair; they give the same price to all buyers. This appears to be the appropriate model for commodity sales to customers with a valid credit card. But for the big transactions involving competition sensitive data, business isn't fair. Some clients are friends and negotiate in good faith but others waste time and money. Consequently, common business sense dictates that a company offer better terms to partners with a good history and that the company extract a surcharge for dealing with unreliable or unknown partners. All the data are encrypted in the Third Way; therefore, a player can offer different terms to different players. Nobody will be able see that this is happening so there are no embarrassing confrontations over the fairness of the terms.

Similar advantages would accrue to other markets supported by a third-way intermediary.

[14] The illustration uses results of software written to demonstrate the third-way using Python technology. In future drafts of this book, the software will be rewritten to use the Java technology discussed in *Part 5 — Tools*. At that point, we can discuss the illustration in more technical depth.

4. Design Considerations

Request Tasks

An implementation of the Third Way will need to use various features of good software design. Some of these can be implemented as “classes” or “types” of software object. A good design provides generalizations of the most useful objects. In this section we encounter one such generalization that is important to and somewhat specialized to the Third Way’s method. It is called a *request task*. A request task is a unit of work as well as a unit of information as we explain here.

Actually, request tasks figured into the previous *Overview*, but it seemed reasonable to defer introducing them by name. For example, in Use Case 1, Figure 5 page 28 we saw a request from the user sent to a system. The user’s request is an example of a *request task* that is issued, immediately executed, and ends when the response comes back. Not all request tasks execute immediately. Some take time to complete and others hang around in the system watching data change and responding whenever they have something to report. So ultimately a request task is just a specification created by a user somewhere and executed on a system.

This definition puts words to something occurring implicitly during the blind agent operation shown in Figure 2, page 24. The blind agent is just a service with fiduciary responsibility towards it clients — the peers in the clique of systems. The clients supply requests to the fiduciary. The blind fiduciary then executes the request tasks concurrently on the federated, encrypted information supplied by the clients.

Although we only need to identify the abstract idea of a request task, it is probably worth pointing out some desirable implementation characteristics. Specifically, the implementation of the request task should include the following:

- Method – the algorithms, patterns, or search statements applied by the Blind Fiduciary during the session. A method is often configured with parameters from a selection specification, which we define as follows.
- Selection Specification – contains parameters that constrain the method so that it finds a small number of relevant results. For any task, there may be multiple specifications and the method is repeated on the federated information for each and every specification. A selection specification might be attached to one or more methods.
- Rationale – Commentary explaining the significance of the result.
- Constituency – who should learn the results? In some cases, the answer is “all” – all members of the clique – but in most cases the constituency for the results should include only a subset of the clique. Specifically, only those peers who contributed information to the result should see the result because they demonstrate a “need to know” by having a piece of the composite result.

To make these desiderata more concrete let's take, as an example, the situation for the Intelligence agencies described on page 22, namely the international comparison of visas to identify potential terrorists, and see how it might apply:

- Method: join tables of encrypted data federated from United States, Great Britain, and Yemen (inner join) matching suspect names with names of visa holders.
- Rationale: names mentioned as suspect terrorists that also appear on a list of visa holders should be investigated immediately.
- Constituency: the country that issued the visa.
- Selection Specification: none

Ok, we threw in a technical term, "join", but hopefully the idea is clear. Anything generated by the above request task is immediately actionable. There is another request task that might be used:

- Method: join tables of encrypted data federated from United States, Great Britain, and Yemen (inner join) matching suspect names between countries.
- Rationale: names mentioned as suspect terrorists in two countries indicate that the countries should exchange what they know about the suspect.
- Constituency: the countries that list the suspect.
- Selection Specification: none

Bailout Points — Use Case 4 Reconsidered

The section entitled *Use Cases* found in Part 1 *Overview of Concepts*, page 11, introduced the use case notation which is a way of writing down design information about the sequence and relationships of events occurring during the operation of a system. In the overview section, there were two types of cases: events between *Users* and their local *System* and secondly events connecting systems in the *Society of Systems*. The overview presented a few aspects of the first kind in Use Cases 1, 2, and 3. There is no reason to develop those cases in this book because the design of any application will be specialized for the application as well as the experience and training of the users. Each application is special.

More design detail can be provided for events between systems within the Society of Systems. These events were treated generally in Use Case 4 that was introduced in the section on *Sessions,* page 30. Now we return to that use case and add an event diagram to reveal features of the Third Way that provide control and protection to the systems.

In realistic applications, the peers show a mutual suspicion and concern about losing control over their information. Automation only exacerbates those concerns. Once a highly automated production process is set in motion, results are found fast. The results, unfortunately are sensitive and confidential. Thus, an automated method can create a lot of trouble fast and trouble engenders fear. Fear scares off potential users.

The Third Way fails unless it can alleviate fear and move the mutually suspicious peers towards mutual cooperation. To foster cooperation, we need bailout points and check points. These features offer control and protection for each peer. They will help in the typical scenario where the peer systems start out afraid of security breaches and afraid of other peers. In this section, we discuss bailout points. In the next section, we will describe checkpoints that limit distribution of results.

The event diagram shown in Figure 21 on the next page marks three important bailout points along the timeline. Bailout points are points in time during a session when a system can stop its participation in the session. Looking at Figure 21 we see two events — "publish session" and "ask to join session" — that were already present in Figure 8, on page 31. At the next event on the timeline, the session organizer publishes a list of session members — the session clique. Then, the proposed members of the clique must decide individually whether to actually join. At this point, they know the identities of the peer systems and have an opportunity to bail out of the session. No system is forced to participate with another system that is not trusted to deal fairly and reasonably.

Cooperation

We assert, without offering proof here, that the ability to walk away at one of the bailout points is a feature that is essential to the development of cooperation between the peers. Proof, such as it is at this time, emerges from game theoretic studies on the evolution of cooperation as discussed in "Commentary on the Theory (to be written)" on page 82.

At this point in time, the peers in the clique know the general subject matter of the session, they know the members and they may also know what general methods the blind information fiduciary plans to apply to the data to discover connections and need-to-know evidence. However, the peers will enlarge the session with additional methods and queries that concern them under their particular circumstances.

This last point is important. Certainly there are general algorithms for discovering information that should be shared. Most important discoveries, however, require some knowledge of the specifics that only one or both of the peers can supply. This point came up earlier in the differences between Figure 1, page 23, and Figure 2, page 24. Let us discuss those differences now by bringing together parts of the two figures in Figure 22 *Comparing (A) Local and* (B) Third-Way Discovery, next page.

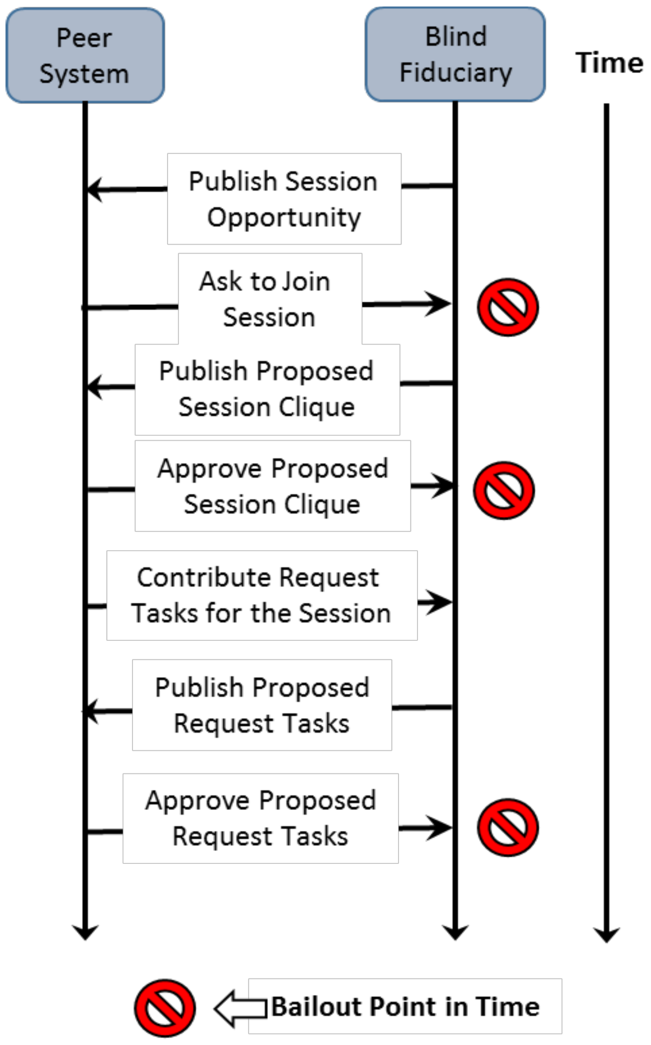

Peer System
Blind Fiduciary
Time
Publish Session Opportunity
Ask to Join Session
Publish Proposed Session Clique
Approve Proposed Session Clique
Contribute Request Tasks for the Session
Publish Proposed Request Tasks
Approve Proposed Request Tasks
Bailout Point in Time

Figure 21. Event Diagram for Use Case 4, Part 4.

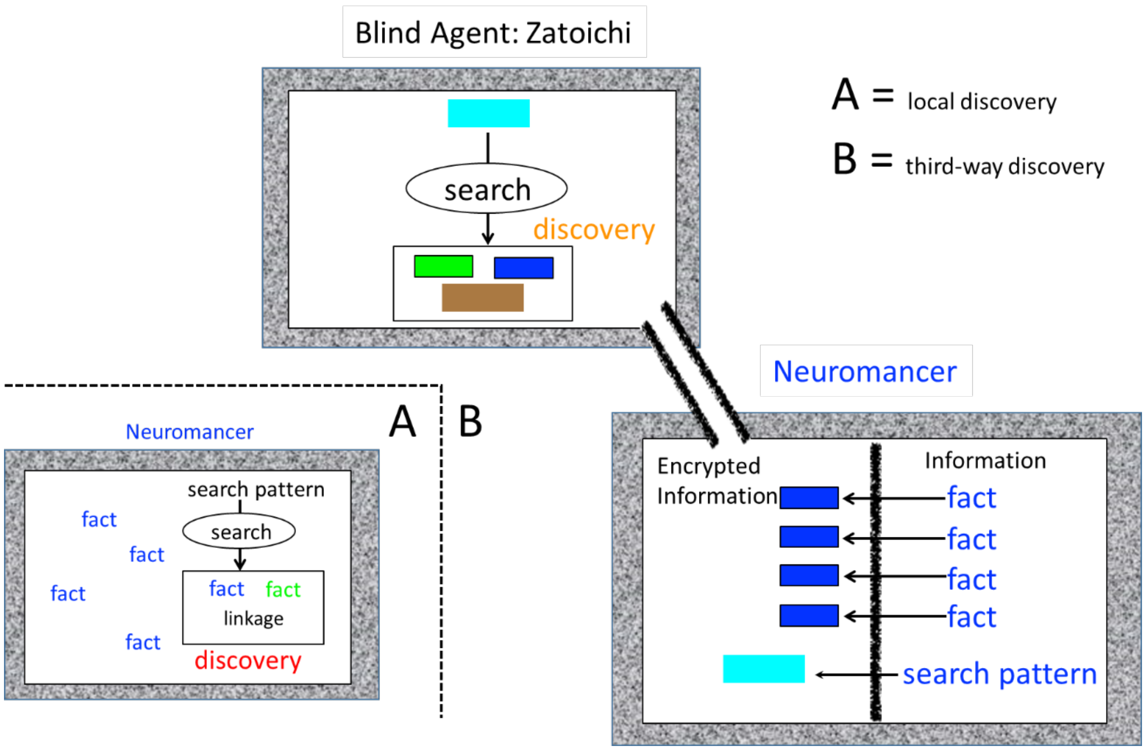


Figure 22 Comparing (A) Local and (B) Third-Way Discovery

When searches are performed on one of the peers, e.g. Neuromancer in Figures 1, 2 and 22, it is valuable to use a specific search pattern or search query. Part A of Figure 22 shows that option. However, in the Third Way, the search must be performed on a blind agent, e.g. Zatoichi, so that the facts are protected from loss due to a security breach in the peer. Part B of Figure 22 illustrates how the search pattern is encrypted and then sent to the blind agent to perform the search. Essentially, this flow of encrypted search information enlarges the scope of the search or discovery process performed by the blind agent.

Let us review this critical part of the process. Under the conventional approach to federating data, the Neuromancer peer had a search pattern that it wished to run on all the unencrypted data. That was the reason unencrypted data was allowed to flow from the Wintermute peer to Neuromancer as shown in Figure 2. You could say that Neuromancer set itself a *request task* and then ran the task itself. Under the third-way approach, Neuromancer sent its request task — search method and search pattern together — to the blind information fiduciary, Zatoichi, which executed the task on encrypted data.

If you look carefully again at Figure 2 or Figure 22, you will note that Neuromancer keeps its search pattern on the secure side of its internal barrier and encrypts the pattern before sending it to Zatoichi. There are two reasons for this step. If the pattern is not encrypted with the same session key as the encrypted data, then the pattern can never match data. So encryption of queries is necessary to the method.

Moreover, it is also true that the encryption of query specifics is highly desirable. For many of the peers, their activities and lines of inquiry are just as sensitive as the information. If someone could steal the queries, they could deduce what concerns the peer has, what it is working on, or what are its methods. Thus, the peers must conceal queries because they reveal the peer's particular circumstances.

Now returning to the timeline in Figure 21, we observe a step when the peer systems contribute request tasks to the session at a time labelled in Figure 21 as "Contribute Request Tasks for the Session". These contributed request tasks are sanitized by redacting from the request those specific parameters that describe the particular circumstances. The blind fiduciary assembles all the sanitized tasks and proposes them to the peers. At that point, the peers can review all the tasks and dispute any one of them and, if necessary, bail out of the session without revealing any of their data.

To help understand why sanitized requests are used in the approval step, let us consider an example. Suppose company A and company B are both involved in the synthesis of complex biological molecules for prescription drugs. If B has an efficient process, A might want to license it. On the other hand, such companies do not advertise which biologics they are working on. That is competition sensitive information.

Instead of advertising for collaborators, the companies can use the Third Way and participate in a session convened to match companies interested in the synthesis of the same biological molecule. The Third Way protects competition-sensitive information but does match companies. The sanitized version of the request task used for this session will not reveal what molecules are of particular interest to the companies. The sanitized request task will only outline how the research on particular molecules will be matched.

Another example can be drawn from criminal and counterterrorism investigations. Agency C and Agency D might want to connect and cooperate if they know they are investigating the same person. Publishing lists of suspected persons is dangerous for two reasons. First, the list may alert the individuals on the list. Second, it would be illegal to publish the list because that impugns the integrity of private citizens without trial. As we all know, there are organizations like WikiLeaks that will go after sensitive information merely to embarrass governments without regard to collateral costs.

A more serious object to omission of the selection specification is based on the fact that the specification may be too broad. The specification might be cast in such a broad manner as to specify the indiscriminate exchange of information. That would invalidate the purpose of the Third Way if it were allowed. Unfortunately, we can't nail down everything at this point in the use case. We will return to this danger in the discussion of checkpoints.

So we have these basic bailout points:

1. Don’t apply to join the session because the proposed session is unacceptable or merely irrelevant to you.
2. Refuse to join the clique after learning who the proposed members are.
3. Refuse to continue the session when the request tasks are published.

These bailout points allow the peers to control what will happen during the session. Obviously, the peers may disagree. The threat of a bailout should bring them to the negotiating table. The negotiation of terms might occur by back and forth proposal and counter proposals submitted to the session manager, the blind information fiduciary. Or, the negotiation may go off line — face to face. The process may be slow but it is preferable to permitting automatic processing that puts security or privacy at risk. It is unlikely that the negotiation would be repeated when the same peer group repeats a session; very likely, the overhead will be minimal when averaged over several sessions.

Checkpoints — Use Case 4 Reconsidered

Checkpoints are places in the use case processing where a peer or a blind agent can hold back data or results. The checkpoints fall into three categories that execute sequentially in time.

Submission Checkpoints. The submission checkpoints implement selection restrictions[15] on what kind of information or which particular instances are shared, in encrypted form, with the blind agent. Every peer should consider whether to send all information or limit information that might conceivably be employed to deduce sensitive information. These checkpoints are entirely under the control of the peer and no part of their operation is visible to other peers.

Distribution Checkpoints. The distribution checkpoints implement restrictions on the distribution of results found by the blind fiduciary to the various peers. These checkpoints are negotiated and approved by the peers during the setup of the session. The blind fiduciary then enforces the checkpoint during operation. It is recommended that any third-way implementation should support two types of checkpoint:

1. Relevance: A result is distributed only to a peer that has contributed information to the deduction of the result.

2. Volume: For every request task, there is a limit to the volume of results. The limit might be expressed in absolute terms or as a percentage of the information inspected by the blind fiduciary. This checkpoint will stop an overly broad request task that somehow escaped attention during the formation of the session (see the last step of the event diagram in Figure 21 on page 63).

The various applications of the Third Way may add other checkpoints useful to their domain.

Fulfillment Checkpoints. These checkpoints operate when the rules given to the blind fiduciary specify that it return only links connecting information items and not the encrypted items themselves. The results are not fully available to any peer until the peers exchange their essential part of the whole result. Each peer then has the control to block the fulfillment of the exchange.

During this final exchange, the transfer is peer to peer. A peer's checkpoint can operate rather subtly. For example, one peer may consider the reputation of the other peer. If the reputation is poor, the first peer may refuse the exchange or hold back part of the information.

[15] *The selection is easily implemented with standard techniques analogous to the projection operation in relational databases.*

Enhanced Security — Zoned Defense

Basic Security

System security is a paramount concern for everyone under all circumstances. In the preceding sections, you've already encountered features that make the Third Way secure:

1. The strong separation of session encryption keys from the federated, encrypted session data.
2. Checkpoints restrict information flows based on rules. The rules may consider access-based security, information volume, and other factors.
3. Encryption of data in use which completes the *encryption triad*. The encryption triad starts with two commonplace security features that protect data and completes the data security protection with a novel third:
    a. *Encryption at rest* — data residing in storage in a system remains encrypted until it is used so that rogue software cannot read it.
    b. *Encryption in motion* — data in transit between systems is encrypted so that a wiretap or man-in-the-middle cannot read data.
    c. *Encryption in use* — in the Third Way, data is encrypted when it is federated for the purpose of finding connections or reaching conclusions. This requires the special agent we discussed earlier, the blind information fiduciary. While information is processed by that agent, it is encrypted so that a covert diversion from the fiduciary will yield only encrypted data.

In addition, the Third Way lends itself to an optional *zoned defense*.

Defense Zones

A Third Way system runs naturally on software designed around data pipelines. By that, we mean that the work can be divided into pieces so that the work-in-progress flows from one piece to another in a sequence until the work is complete. When software can be designed as a pipeline, different parts of the work can be allocated to individual host computers in isolated locations that we call zones. The zones can be used to strengthen the security.

The zoned option becomes a defensive measure when we introduce these design elements:

- Each zone runs a restricted set of software functions. The host in one zone does not have enough software to run the full service nor should it have the extraneous software normally included when the machine is delivered. Cyber attackers often exploit defects in delivered operating system software (zero-day exploits) and a simple way to foil such attempts is to clean out unnecessary items.
- Zones are connected by a network but the network connection is protected by a firewall with two components: first, conventional protection based on ports and network-packet inspection and second, a protocol filter that inspects each network message to verify that the message is legal and well-formed in the context of the applications allocated to a zone. With a protocol filter, an attacker will find it difficult to connect with malware even if the attacker has found a

way to install malware in a zone. A protocol filter also blocks attempts to crash or take over a computer by sending a malformed message[16].

Many of the design elements of the Third Way are essential. Zoned defense, in contrast, is recommended but not required. Because this defense is an option, we will give only an overview, but we will illustrate it by a reference design that can serve as a starting point.

The reference design adds three defensive zones to a peer system in addition to any other defenses that the peer may install. For the blind information fiduciary, it adds two zones. In the reference design, the zones were chosen so that each zone holds one of the three encryption keys that are required by third-way operations. No intruder on a host in a single zone can get at all three keys. Furthermore, an intruder who breaks into two hosts in two zones cannot combine stolen keys because the firewall's protocol filter prevents communication between hostile software in distinct zones.

Consider what this means from the intruder's perspective. The intruder must install malware in three zones but only the outermost zone connects to the outside world. Any malware in the outermost zone has trouble gaining access to the next zone for two reasons. First, it doesn't have the key used in the next zone because it is not available in the first zone. Second, any unexpected or malformed message between zones is blocked by the protocol filter. This kind of multilayered protection will certainly be needed in the threat-laden cyber future.

The strength of a zoned cyber-defense invites a comparison by analogue with armor for military vehicles. In World War I, the armor plates protecting a tank or battleship were a thick layer of steel. Rapidly, shaped warhead charges were devised that could penetrate thick layers of steel. Initially, there was a race between offense and defense, with armor becoming thicker and heavier until the protected machine became too heavy to maneuver and too much of a fuel guzzler. But defense recovered its edge when *composite armor* was developed. Composite armor has multiple layers of different materials that provide excellent penetration resistance — a single layer of material could only achieve this level of resistance if it were impractically thick and heavy. This analogue justifies calling the zones a composite-defense for cyber-security.

In the reference design, the blind information fiduciary has only two additional zones because this system has access to only two of the essential encryption keys. Figure 23 on the following page illustrates the three zones added for a peer. The reference design for the blind information fiduciary is the same except the innermost zone is deleted. As explained earlier, a peer's software is divided during the design phase and then allocated to host computers located in multiple zones and connected by the flow of message data.

Information passes between zones as messages. This statement applies to most systems, even those that appear to allow a constant flow of messages. In order to send a steady flow of information, a computer breaks the flow into a stream of discrete messages. The approach of streaming messages has the advantage that we can wrap a message in other information, creating essentially a payload in an envelope with addressing information on the envelope. When the messages must pass through network

[16] We are thinking here about attacks involving buffer overflow or SQL injection as well as random probes using high-frequency messaging to look for a zero-day software defects.

layers or zones, we can take one envelope and wrap it in a second so that the address is appropriate for the destination of the message. This approach has been fairly standard since the evolution of the

Open Systems Interconnection (OSI) model[17]. You will see that the Third Way uses envelopes, but it not only annotates the envelope with addressing information, it applies encryption and digital signatures.

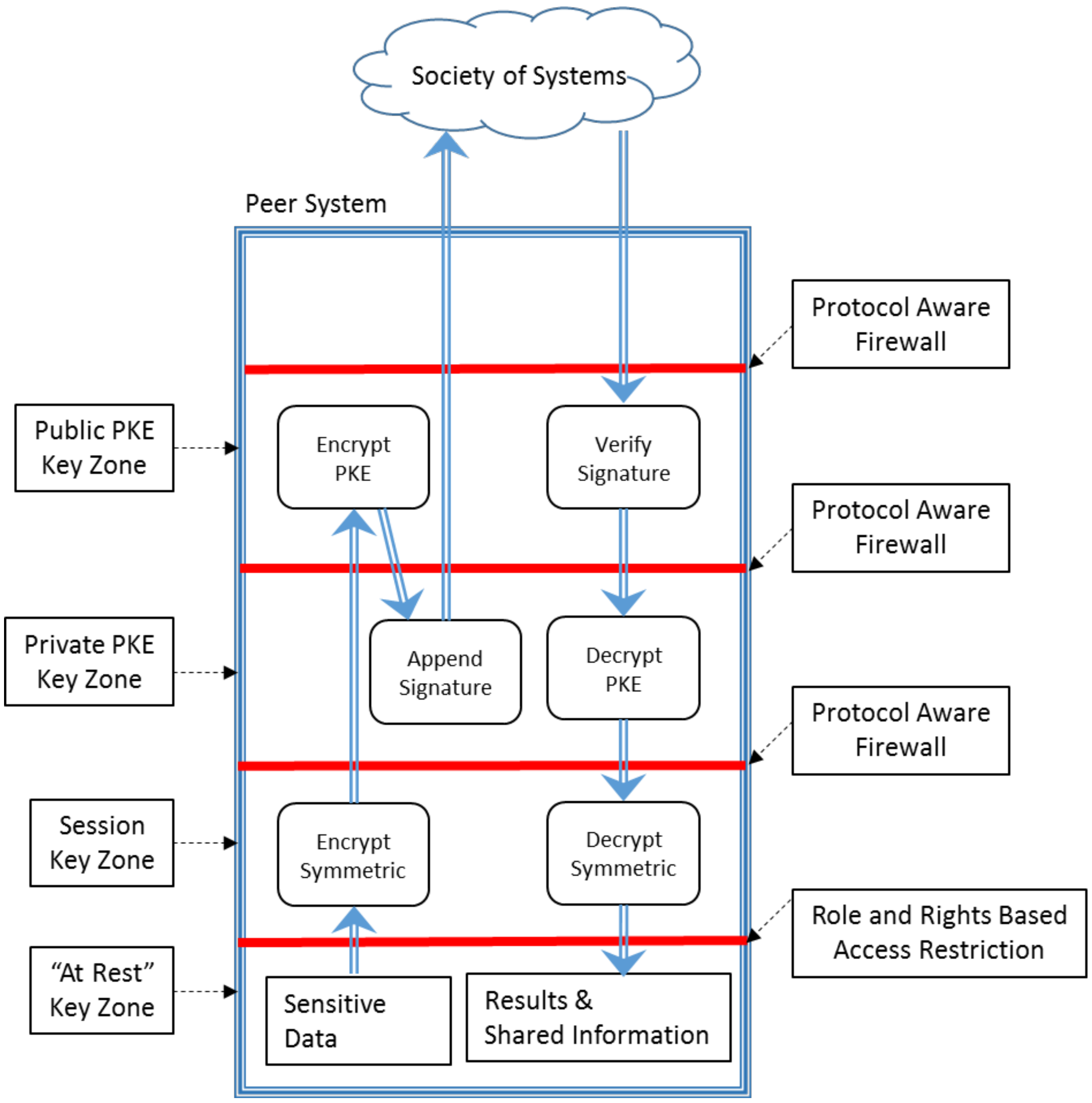


Figure 23. The Reference Model for Defense Zones

[17] See for example https://en.wikipedia.org/wiki/OSI_model

Let’s consider the efferent (outgoing) message flow starting in the session key zone in Figure 23 and then following that flow from there. Here are the steps beginning with the originator of the message:

1. Information is encrypted in a session key zone so that it can only be read in another session key zone. Leaving the session key zone, messages go to the public key zone.
2. A message passes through the public key zone where the public key of the recipient system encrypts the message; therefore, the message can be delivered successfully only to the intended recipient system. Leaving the public key zone, messages go to the private key zone.
3. In the private key zone an efferent message acquires a message digest and a digital signature. The signature uses the private key of the sender and the key also encrypts the message digest. These steps will prove the identity of the sender to the recipient as well as verifying that the message was not altered in transit. The message then leaves the sender’s system bound for the recipient.

The recipient of the message now completes the transfer of the message with these steps:

4. Afferent (incoming) messages enter the public key zone, where the public key of the external peer system is used to verify the peer’s identity and the integrity of the message. The afferent message then moves to the private key zone.
5. Arriving messages are decrypted in the private key zone. In the unlikely event that the message was not intended for this peer, the private key does not match the public key and the result is a damaged message that should be rejected. Afferent messages then travel to the session key zone.
6. The arriving message is decrypted in the session key zone by the session key.

In practice, production designs will elaborate on this reference design.

Black/White Composite Defense

The reference design described above complies with Kerckhof's Principle: the security is based on hiding keys and not hiding the algorithm. In fact, the reference design could be implemented by open source software[18] that might be strengthened by the talents of many independent code reviewers. An open source implementation would follow what we might call a White Box metaphor. Everyone knows what is in the box.

On the other hand, many put faith in a Black Box metaphor: software is closed-source, propriety and secret. If you have that faith, then it will make you feel more secure if you add to the reference design some features that apply closed-source security defenses. When black box and white box software are combined we might say we have a *Black/White Composite* defense.

Opinion

The value of black or white software is hard to evaluate convincingly because it depends so much on the implementation of the software and on careful execution during the deployment phase. However, a black/white defense may make sense because the two types have different, complementary risk factors. Open source software can be reviewed by many sympathetic, white-hat experts who can find flaws before the software is released to the scrutiny of attacking, black-hat experts. On the other hand, closed source software can thwart an attacker if its internal security tricks are hard to derive by stressing the system or applying big data analysis to a very large set of probes against the security layer. Both open and closed source software have their place. At the current time however, the arguments for open source software fall short because there simply aren't enough experts with time on their hands to test open source software for security flaws. But, the same argument can be applied to closed source software implemented by understaffed teams driven by an unrealistic schedule.

[18] If this term is unfamiliar please see https://opensource.com/ and/or https://en.wikipedia.org/wiki/Open-source_software .

Encrypted Discovery of “Need to Know”

Introduction

The Third Way promotes the exchange of information on a need-to-know basis. How do we determine need to know? Let’s consider the conventional approach first. The term “need to know” comes from established security practice. In conventional security practice, “need to know” is a decision reached by a security officer. A human must examine the facts of the situation and determine whether a particular party has a need to receive access specific items that are classified or sensitive. There are obvious drawbacks to employing a human in this situation:

- The through-put of a human is not adequate for modern data rates and data volume.
- The human may err by finding “need to know” when it is not justified or failing to find it when there is justification.
- Humans are often fallible, corrupt, or careless and the sensitive data is at risk when it is shown to the security officer.

The through-put problem has only grown with the rapid expansion in the volume of sensitive, stored information. There aren’t enough people to enforce a need-to-know access policy; consequently, the vast amount of data that is vulnerable to theft today is actually less secure than in the past. The target is broader while the defenses are less deep. Thus, there is ample justification to automate the need-to-know decision to handle large information volume.

To be sure, there are drawbacks to the automation of the decision:

- A human will improve with practice and experience whereas the automated decision operations considered to date do not. Future research may alter this situation.
- The decision algorithms must be implemented with a limited set of comparison operations.

The last drawback is a consequence of the fact that the comparisons of secrets must be performed on encrypted information to preserve the high degree of security afforded by the Third Way. However, there is some experience that points to the broad capability of the encrypted comparisons. At MIT, the CryptDB project constructed an encrypted mirror server for a large, heavily used relational database. A feature of this academic database installation is that all queries are recorded for later study. Such queries were written in SQL, a common query language. Each query controls a fairly general decision mechanism, the database’s query processor, to make a selection of particular records from the large database. That selection is equivalent to determining which information should be merged and made available for sharing. The selection is exactly the capability needed for the blind information fiduciary. Consequently, experience with the CryptDB project should be applicable to our subject.

The results of the CryptDB project demonstrated that encrypted queries could be matched to the proper encrypted information in 95% of the sample cases[19]. It seems likely that the success rate of 95% might be increased if it were permissible to reformulate the 5% of difficult cases so that the queries better

[19] http://css.csail.mit.edu/cryptdb/
http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb.pdf

match the encrypted system. In addition, we can point to one commercial system that used encrypted discovery, IBM's Anonymous Resolution[20]. In view of these projects plus the three sample applications describe here in Part 3, the reader should be assured that encrypted data can be used for a discovery process producing need-to-know information.

A query processor for SQL operates by making a series of field by field comparison operations. The set of available comparison operations must be adequate to allow the operation of the query processor. The question then is: are the needed comparison operations workable on encrypted information. The general answer is yes. This chapter provides a more detailed look at the comparison operations.

Comparisons

Equality. It is possible to compare two encrypted records if and only if they are encrypted with the same algorithm and the same encryption key. When that is true, values that are equal before encryption are equal afterwards. That observation is at the heart of the decision process in the Third Way.

Set Membership. The equality test can be used to provide a set membership test. The range of the set is first converted to an encrypted range. The encrypted value can then be compared to the encrypted set. If there is a match, then the unencrypted value must match the unencrypted value of one member of the set. This comparison operation is more important than it seems by its name. Matching the names of people may require matching one name with a set of aliases, alternate spellings or nom de guerre.

Mutual Set Membership. If we have two sets, then we can encrypt the values of each range and perform the usual operations of set-intersection and set-union. The result of these operations is an encrypted set. That would not be useful for the decision process; however, we can count the members of the set intersection and that number, an unencrypted scalar, could be used to reach a binary decision about "need to know".

Relative Value. If we have numerical values and conceal them with order-preserving encryption, then we also retain the use of standard comparisons such as less-than, greater-than, less-than-or-equal, etc. These comparison operations yield a binary, true/false result.

Range Overlap. Any range of values (e.g. 3.6 – 7.1) can be concealed with order-preserving encryption and compared to another encrypted range. If there is an overlap in the encrypted ranges, then there must be an overlap in the unencrypted ranges. Thus, we have a binary result that can be used for a decision. If the overlap exists, it may provide useful ancillary information, although the actual range remains concealed.

[20] http://www-03.ibm.com/software/products/en/anonymous-resolution However, since these words were written, IBM has withdrawn support for the product. This link may not work. Information may still be available at archive.org from snapshots of earlier IBM websites.

For each comparison method, there is a corresponding result that supports the decision. If you look closely, there are four possible results of the comparisons just listed:

1. Yes/no, that is, a binary result. The equality, relative-value, and set membership operations yield a binary result.
2. A binary result paired with an unencrypted integer score (e.g. 5 or 27 etc.) The set intersection operation yields the number of set members in the intersection and a binary value stating whether the intersection is or is not empty.
3. A binary result paired with a numerical range concealed by order-preserving encryption (e.g. 4.32 to 7.89). The range-overlap operation returns such a result.

It is very cumbersome to write “concealed by order-preserving encryption”; therefore, we simply say “concealed”.

The choice of words to use here is controversial. The technical issue is whether or not an order-preserving algorithm that uses an encryption key should be called a form of encryption. Some in the security community are willing to use the term.[21] On the other hand, the strength of the protection offered is much weaker than that provided by more common encryption, which does not preserve order. It seems important to bear this distinction in mind when building a secure system. For that reason, we use the term “concealed” in preference to “encrypted with an order preserving algorithm”.

We will also mention another kind of result that does not allow decision making:

4. An encrypted computed value (e.g. 6.28318)
   This last type of encrypted result can be generated by a number of operations understood under the heading of “homeomorphic encryption”. Some applications may wish to compute such a result and return it as useful ancillary information. However, the results from homeomorphic encryption operations cannot contribute to the need-to-know decision because that decision is reached at a location where the encryption key is inaccessible.

Eventually sharing is a question of a yes/no decision. The final decision is usually the result of a series of preliminary yes/no decisions that are generated by comparing corresponding encrypted fields in the information records.

The last method-type, type 4, can play no role in a decision because the decision is reached by an information fiduciary that does not have the encryption key available. Nevertheless, if the decision is made to share, then the fiduciary might include the encrypted computed result as one of the shared products.

The results that support a decision are produced by low-level operations that work on individual fields in a data record. Real information records have many fields exhibiting complicated relationships. Another way to say this is that real records contain structured information. How can we work with real records? The answer hinges on standards.

[21] http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb.pdf

Standards

For there to be any successful exchange of information between information sources, standards must be created and followed. At this point in time, the way that text and numerical information in an information record is arranged and represented in the one computer can present an obstacle to understanding the record in a second computer. Standards ensure that each record can be understood in any computer belonging to the community.

Much of the world's information is not stored in standardized form. Not surprisingly, there are major efforts underway to enable machines to understand non-standardized natural sentences and similar artifacts from human verbal and written interactions. The achievement of this understanding is in its infancy and is changing too fast to discuss here. However, we note that, when successful, these new free-text processing methods can be used to extract knowledge and place it in a standardized format. Therefore, the third-way approach can be used with free text so long as the free-text can be understood at all.

There are many competing standards efforts. That multiplicity has always been the *Achilles Heel* of standardization. For this reason, we mention only one standard, the *National Information Exchange Model* or *NIEM*[22]. This standard applies to XML (eXtensible Markup Language) documents. The XML document format is successful in controlled environments that can take advantage of well tested XML tools for record creation, manual editing, automatic proofreading, verification, and parsing. XML is not the most efficient format for storage and transfer however. Consequently, many organizations use a combination of XML and JSON (JavaScript Object Notation) so that they may combine the tools available in XML with the compact, efficient representation method of JSON. There is no reason to go into these topics in more depth here but it seems worth mentioning their importance.

Structured Information

Introduction

Information in a computer is represented in a standard, structured, format. Of course, a computer can also handle free text and perform a credible job of formatting it for printing or web pages. However, we assume that the essential information has been extracted from the free text and lodged in refined, structured information.

Let us assume the information is stored as a set of records, **R**. These records may not be standard but we must assume there is a transformation from each record **R** to a standard record **S**. Lastly, we assume an encryption procedure that can be applied to the variable field values of **S** so that it transforms it into a standard, encrypted record **∑**. These transformations are invertible so that we can eventually transform encrypted shared results to their unencrypted forms. We can summarize the operations as:

$$R \rightarrow S \rightarrow \Sigma$$

Where **R** *is a nonstandard record,* **S** *is a standard record* **∑** *and is an encrypted standard record.*

Query Languages

[22] Please see http://www.niem.gov

We have seen that decisions are based on operations that compare two values or compare one value to a set or range of values. The standard record format provides a structure for the values, but the decision operations operate on value fields of the structure. How does that work? It turns out that this is very hard, but long ago the task was made easy by query languages. An analyst or other user states a decision criterion in a nonprocedural language statement and then the computer does the heavy lifting behind the scenes to convert the decision criterion into a procedure. The computed procedure specifies a series of comparisons between individual field values in the record and logical operations that combine the results of each step.

You have a choice of query languages but it pays to use a language that is widely written and read by people. For queries, that language would be *Structure Query Language* or *SQL*. There are other languages for specialized information disciplines but we shall concentrate only on this widely known one. As this is book is being written, there is only one published study that indicates the success rate for converting SQL operations on unencrypted records to the corresponding operations on encrypted records. This study was discussed already in the introduction to this chapter. For real historical queries in SQL, an MIT team achieved a 95% success rate[23].

Comparison of Structured Information Records

Just as we can apply arithmetic and logic to numerical or Boolean values, we can apply many of the same methods to encrypted, structured information records.

If a standard encrypted record $\sum_a$ deriving from a source denoted by the subscript **a** has a series of value fields, (ζ_{a1}, ζ_{a2} , ζ_{a3}, …), we may compare the record with a second standard encrypted record (ζ_{b1}, ζ_{b2} , ζ_{b3}, …) deriving from a source denoted by the subscript **b** . The comparison is performed field by field; that is, we compare ζ_{aj} with ζ_{bj}, for all values of the index **j**.

Unstructured Information

The world is flooded with unstructured information: text messages, written documents, captured speech, etc. We have pushed aside the consideration this information because it is outside the scope here but there are definitely ways to treat such information. For that reason, we expect that unstructured information will be stored and then processed to yield structured information. We can then work with the derived structured information. In fact, one of the applications described in Part 3—Applications works with unstructured text (see ”A Context Free Illustration” page 49.)

[23] http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb.pdf

5. Tools (partially written)

Currently, the outline for this chapter looks like this:

1. Packages — description of Java SE modules that are now currently available and documented on line[24].
2. Sample Applications — description of the demonstration software used during the development of the tool components. Additional applications will probably be developed before declaring this part complete.
3. Servlets — these are specialized Java modules running on the information fiduciary and allowing users to interact with the fiduciary via a secure web browser page. These have yet to be developed and documented.
4. EJB — Enterprise Java Beans, are containers for Java code that allow the assembly of applications from larger more intelligent pieces. They have not been developed or documented.
5. Repository — this is an online location where code and documentation is stored in a version controlled environment. Location is TBD

At the moment only the package description is available.

Implementing a Third Way Software Application with the Java Package Library

[24] See: http://www.wwnsoftware.com/OpenBEDM/javadoc/index.html

6. Supplemental Topics

Overview of Kerckhoff's Principle, Encryption, and Keys

This chapter will explain what you need to know about encryption in order to understand the Third Way. The chapter just covers the encryption methods used in the Third Way and even then it neglects technical detail. However, the reader who is interested in encryption can turn to any of many good texts, for example, Ferguson, Schneier, and Kohno 2010[25].

Keckhoff's Principle

At the dawn of modern security practice in the 19th Century, Auguste Kerckhoff published two papers setting forth 6 principles that any secure system must follow. Jump to the 21st century and we find that 5 of his principles embodied in modern systems. We can safely set those aside. One of his principles, however, remains important for understanding data security. The principle can be paraphrased this way: *information must be encrypted in a way that it is secure even if the opponent knows everything about the process except the encryption key*. Today the term Kerckhoff's Principle[26] is applied to this one important part of his original set of principles.

Cryptographers have developed encryption algorithms that are for all practical purposes unbreakable. Implementation of the Third Way uses two classes of these encryption algorithms and our purpose here is to introduce those algorithms for the non-specialist. All the algorithms are published. Their security rests on the idea that the algorithm ensures security even if it is known. That is Kerckhoff's Principle in practice.

How can we be so confident that an algorithm is unbreakable? To put it into the terms of an old question, is there a lock that cannot be broken?[27] We feel these algorithms will not be broken because it requires a slow, expensive application of brute computer force, which may or may not succeed. Why would an attacker bother when there are other ways to defeat security? Frankly, security can be most easily defeated by stealing the key. If your system is attacked, your encryption keys are a major prize for the attacker. Your users must employ these keys in their work; therefore, a common method of attack impersonates a user and then makes use of an encryption key with criminal intent. The main outcome is that today's systems are insecure because they cannot stop all the ways to steal keys. That is one of the main motivations for advocating a Zoned Defense as we have argued here. It separates keys from each other and from users.

---

[25] "Cryptography Engineering", Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno, Wiley Publishing, 2010.

[26] See Wikipedia: https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

[27] For an interesting discussion of "perfect-security" as applied to locks, you may be interested in this web reference: http://99percentinvisible.org/episode/perfect-security/

To round out the picture, we should mention two fairly esoteric concerns about encryption security. The first concern is a defect in the system software that creates new encryption keys. If the attacker can modify that part of the system, they can cause it to generate keys that the attacker can later discover using only a modest amount of effort. The best systems generate keys from a physical random source known as the "source of entropy". The physical system, e.g. a radioactive source and a Geiger counter, cannot be remotely hacked.

The second esoteric concern is quantum computing. The claim is that quantum computing can break the public key encryption system discussed below. The claim is based on the idea that there are equations in quantum physics that describe nature but cannot be computed in finite time. Yet, nature still happens in real time! Quantum computing turns the tables around and finds a quantum-mechanical equation that matches the software problem, e.g. deriving a private key from a public key (see below). A quantum computer is essentially an experiment on a properly initialized quantum system. The quantum system is initialized and then the subsequent measurement of the system yields the answer. It sounds simple phrased this way but there is no cause for immediate concern because quantum computing has yet to be realized at a useful scale.

Symmetric Encryption with a Single Key

Symmetric encryption uses a single key for both encrypting a clear text document (plaintext) to an encrypted document (cyphertext) and for the reverse process: decrypting the cyphertext to reproduce the plaintext. Figure 24 illustrates the use of symmetric encryption.

Symmetric key encryption has two applications. It can encrypt data as it rests in storage so that an intruder who gains physical access to the storage cannot then read any of the data found in storage. Secondly, it is also used to protect data as it moves over the network. A side channel that diverts the information in transit acquires only encrypted, useless data. A good example of this encryption method is the *Advanced Encryption Standard* (AES) approved by the *National Institute of Standards and Technology* (NIST).

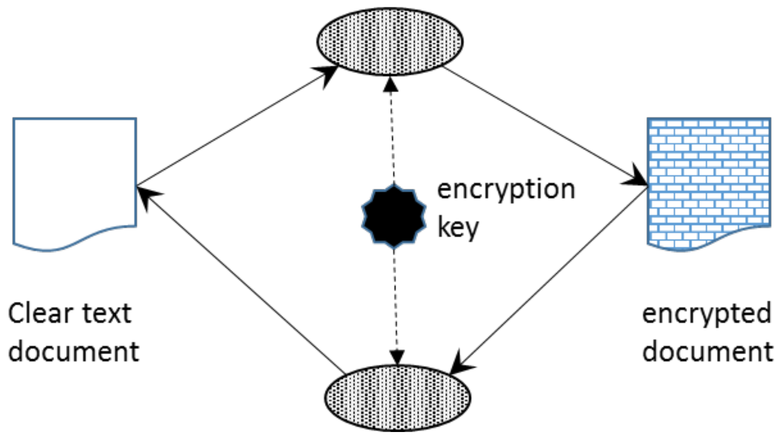


Figure 24 Illustration of Symmetric Encryption Using a Single Key

Public Key Encryption

Public Key Encryption or PKE uses two different keys. One key operates the encryption transformation and the second operates the inverse, decryption transformation. PKE is used in the following way. One of the keys is kept as a secret; it is the private key. The other key is given out as public knowledge. This is the public key. To see how this works, imagine that system DE wants to send a text document to system UK so that it is protected by encryption in transit and so that the UK system and only the UK system can decrypt and read it. The procedure is illustrated in Figure 25. System DE encrypts the document using the public key of System UK. It then sends the document. The encrypted document can only be decrypted with the private key of UK, which UK holds securely as a secret.

You should be wondering at this point how the sending system DE can be sure that the key it believes belongs to the system UK is actually held by UK and not by an imposter. There are a couple of answers to that. There is an obvious manual method. A representative of DE can physically visit UK and obtain UK's public key on a thumb drive. But a manual method for conveying keys is impractical for most applications and it is only used for extremely sensitive communication. In practice, we rely on the identity management information fiduciary that we said earlier is an implicit member of the society of systems. That phrase is quite a mouth-full so there is a shorter name. The public keys are distributed by a *certificate authority* that vouches for the identity of UK.

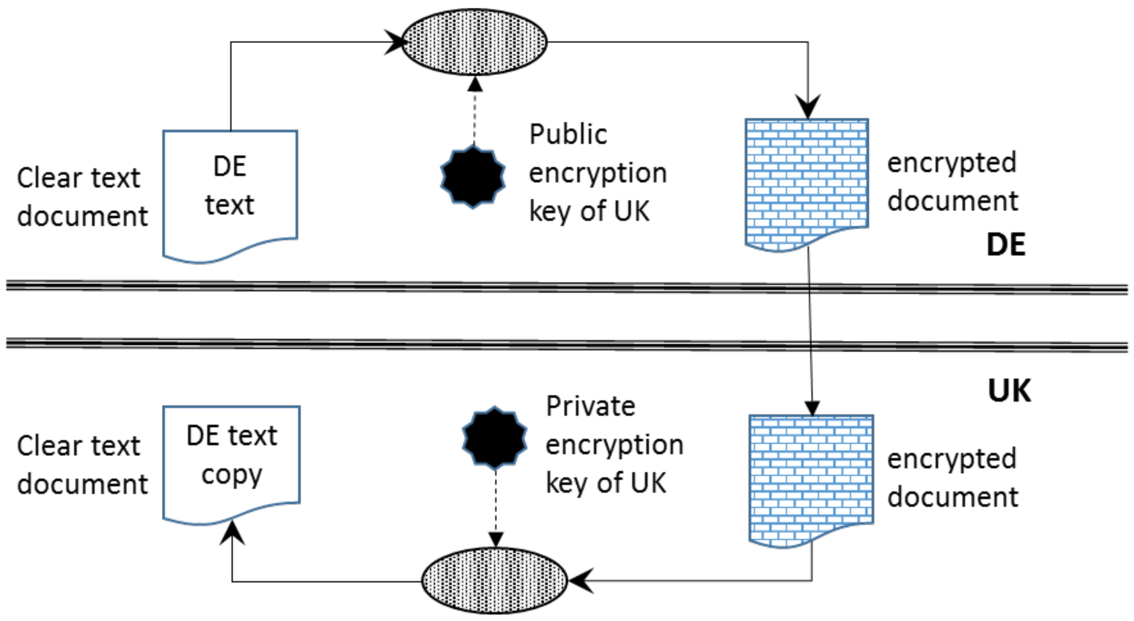


Figure 25 The Use of PKE to Transfer a File Safely from One System (DE) to a Particular Recipient (UK)

The procedure in Figure 25 illustrates a file transfer with PKE. Using PKE gives us two useful results:

- PKE can deliver a document safely to one specific recipient. All others cannot decrypt it.
- PKE can be used to digitally sign a document or file.

To explain the second property, we need to mention algorithms that produce a message digest. A message digest is a short document that is derived from the full document by a mathematical method that appears random. But the method is actually deterministic because the same identical document always yields the same identical digest. What makes this interesting is that a modified document produces a different digest however minor the alteration to the document. Using the message digest, an author or sender can apply a digital signature to a document or file. A digital signature guarantees two things: that the content of the document is unaltered and the signature was issued by a specific, known party.

A digital signature works like this. The author finishes the document and computes the message digest. The signature consists of the message digest encrypted by the private key belonging to the author. The recipient can then verify the signature found on a copy by first creating a new message digest, then decrypting the signature, and then comparing the two. If the two are not the same, the document was not signed by the author or the document was modified after it was signed. If the two are the same, then no party other than the original author could have signed the document because no other party has the private key.

To summarize, as long as the private key remains a secret known only to a document originator, then public key encryption brings several important security benefits[28]:

Authentication — since the individual's unique private key was used to apply the signature, recipients can be confident that the individual was the one to actually apply the signature.

Non-repudiation — since the individual is the only one with access to the private key used to apply the signature, he/she cannot later claim that it wasn't him/her who applied the signature

Integrity — when the signature is verified, it checks that the contents of the document or message match what was in there when the signature was applied. Even the slightest change to the original document would cause this check to fail.

Confidentiality — because the content is encrypted with an individual's public key, it can only be decrypted with the individual's private key, ensuring only the intended recipient can decrypt and view the contents

Other Algorithms

Symmetric and Public Key algorithms account for most of the functionality of security software. However, there are a few other algorithms that should be mentioned.

Hash Function — This type of function converts a sequence of letters or numbers of any length to a sequence of fixed length. We are only interested in a subtype of such functions known as cryptographic hash functions. For these functions, the output is seemingly random whatever the input. Moreover, a hash function has no inverse function. Once hashed, the sequence stays hashed. The terminology has evolved over the years and the message digest function and hash function work the same way. However, the first message digest algorithms like MD5 were not sufficiently secure so they were

[28] *This list of benefits is copied verbatim from this web site: https://www.globalsign.com/en/ssl-information-center/what-is-public-key-cryptography/*

replaced by better algorithms that are called hash algorithms. Today, only the better hash algorithms such as SHA-1 and SHA-2 are used to create a message digest. So you can use the terms hash and message digest interchangeably today. Just keep in mind that older message digest functions are not suitable today.

One-Way Encryption — This algorithm uses a key that encrypts a document in the sense it cannot be read because there is no inverse algorithm. Even with the key, nobody can read the document. This property is occasionally useful; for example, it was adopted for the product “Anonymous Resolution” described below in the chapter entitled “Similar Approaches.” This algorithm is often described as “hashing with a salt”. If you encounter the term, here is how to unpack it. The “salt” is the same thing as the key. The key or salt is appended or prefixed to a document and then the slightly longer document is subjected to a hash algorithm. The result is seemingly a random sequence but one that is determined by the original document combined with the key. But, there is no way to go backwards; moreover, the same document processed with a different key yields a completely different result.

Order-Preserving Encryption — The accepted encryption algorithms yield seemingly random output. Now suppose we start with a set of values that are arranged in a particular order. For example, the sequence 1, 3, 9, 15 is in ascending order. Encryption will produce a sequence of values with no predictable order. This is normally good unless you have an application that wants to compare encrypted values and determine the order of the unencrypted values. For this purpose, there is a class of algorithms that uses an encryption key to remap a sequence to a different sequence with the same order relationships. Arguably this is not encryption. Although you will see the term order-preserving encryption, we prefer to call this a “concealment algorithm.” It is still valuable to conceal values but one shouldn’t claim protection at the same level as symmetric or PKE encryption.

Trust in Software: What You Should Know

The Third Way recognizes that you can't trust each and every person, not even trusted, cleared insiders. However, we have been assuming along the way that you can trust software. However, trust in software puts faith in people, lots of people, because software is a handcrafted product. The potential for betrayal is inseparably bound to the almost limitless functionality of computers that can be used for benefit or harm.

You must at least know this much about software: the stuff itself is dangerous and must be handled with care. In this appendix, I cannot give you expert knowledge — I lack it myself. But I can convey the perspective of a computer-science flâneur, someone who has been places, seen things, and has something to say about them.

Software Maladies: Congenital, Acquired, and Inflicted

Congenital Maladies. A congenital software malady is one that reaches you direct from the vendor or programmer who supplies your software. Fresh out of the box or downloaded as a file, your system software or application very likely contains a defect that can be exploited. It is popular to call these "zero-day" exploits. Both criminal organizations and governments search for these because they enable a cyber-attack on a computer system. There is even believed to be an open market where "zero-days" can be purchased.

Many of these defects are unintentional and not known to the vendor. For example, many zero-day defects feature a buffer-overflow event during the attack. During such an event, data supplied by the attacker is allowed to write over active software code, changing its behavior. A large fraction of contemporary software is potential vulnerable to buffer-overflow because it was written in a language, such as the popular C++, that makes no effort to prevent buffer-overflow. In the past, advocates of C++ have argued that efficiency is paramount. If you don't check the code while it is running, it will run faster. However, for the future, safety from cyberattack should be elevated to a higher priority.

Other congenital maladies, however, are caused by insecure features added intentionally by the vendor or developer. These have a purpose for the vendor or developer and are supposed to be hidden. Eventually, they will be discovered and exploited by attackers. For example, in the early years of the Unix era, an operating system would arrive with a "wizard" account enabled. That account could take over the machine. More recently, Microsoft would leave a port open for the same purpose. Is that era past? It should be.

More of concern today is the software written or modified by software employees on staff who work covertly for a second employer. These two-employer software developers add "zero-day" defects in one vendor's software for the benefit of another party. The period between the deployment of the intentional defect and its discovery can be very long, particularly if the exploit is used sparingly by its sponsor.

Acquired Maladies: An acquired software malady is one that infects a system after it is installed, tested, and running. These infections are acquired from the boot-sectors of removable media, from email attachments, or clicks on special links on a web page. These infections can be spread by social engineering — that is, by giving an unsuspecting person a plausible story to convince them to click or insert media.

The susceptibility of a computer to acquired infection derives from its features more than unplanned defects. The modern computer effortlessly launches software written in a variety of scripting languages with the notion that we want dancing bears on a birthday card or interactive web pages. Also, scripting languages run the background processing and maintenance of the system so they are, to a degree, unavoidable.

Acquired infections might be prevented by modifications to operating systems that would “sandbox” instructions coming from removable media, email or the web. To date, no standard operating system has such a defense, but you may come close by adding software from various third-party vendors. It would be worth the time and expense. Also worth discussing is a “white list”, a method we explain below.

Inflicted Maladies: An inflicted malady is caused by a change to hardware or software implemented by an insider with detailed knowledge of your system. You should put policies in place that limit the damage a single person can cause. One might hope that the NSA, at least, has learned from having granted powerful privileges to its contractor, Edward Snowden. Then again, perhaps it is hard to actually implement this precaution for several reasons. First, you would need additional staff and an increased budget. Second, you might want to hire an outside consulting firm to check for gaps in policy and implementation. That means more money from the budget. The bottom line for decision-makers is that the PowerPoint bullet awarded for this achievement won’t seem justified by the costs it incurs. And who plans for future losses or for loses that go undetected? It’s the PowerPoint presentation that wins next year’s funding so security takes a back seat more often than not.

Restricted Systems

Technically, it is possible to make an installation where it is difficult for a software malady to spread, and the malady cannot cause widespread damage. These are restricted systems — minimal software configurations that do a few things but not the many things that computers are expected to do. Unless the infection can thrive on the sparse menu of installed software capability, it will die. Different operating systems offer such restricted modes of operation under different names. Unix has its restricted shell (rssh) and Microsoft has *limited user accounts.* The strength of protection offered by these options can vary. But first let us outline how a restricted mode works.

Software runs on behalf of someone – an individual person or a role such as sysadmin (short for system administrator, of which there may be several). That person or role verifies their identity and, if successfully verified, is allocated one process. That one process then launches and controls all the others. The first process has a certain level of privilege in the system and access to resources. The processes that the first one starts inherit only the privileges of the first. So in the weakest implementation, a restricted account simply has restricted privilege. Unfortunately, malware tries and often succeeds in raising the level of privilege beyond what was initially intended. A stronger implementation needs more protection.

System managers can take the next step to restrict a system by creating a “sandbox” for the initial process. Inside the jail the process sees only a portion of the computer’s storage, its “sandbox”, and may only execute software found there. This approach works only if the jail contains a minimum essential complement of system software. The trick is to provide just enough and nothing more. For the trick to work, a subset of the full system’s files must be copied into the restricted jail area. For the Unix system, more details can be learned by searching for information on the “chroot” configuration options associated with the restricted shell “rssh”.

The preceding paragraph illustrates the difficulty of the topic: any discussion has to go down deep technically to fully describe the situation. Faced with that task, we will jump directly to the necessary conclusion. To operate safely, it is not sufficient to verify user access to the system. Malware operating under that user’s authority can create damage. It is necessary to limit the capabilities offered to a user to just those capabilities necessary to do their job. Unfortunately, that is a high bar to get over. Modern users expect capabilities like JavaScript, Adobe Flash, Microsoft Office scripting, and a host of other sophisticated software features that may host a software infection.

There is no perfect way to secure a single machine or homogeneous cloud of computers. For that reason, our approach advocates a strategy of defensive zones that were described in the chapter entitled “Enhanced Security — Zoned Defense” on page 68. An organization that follows that strategy partitions the flow of the information and allocates particular functions to individual restricted systems that can perform only their assigned function. In addition, the restricted systems are separated with firewalls configured to restrict network traffic to messages that follow a limited set of approved protocols.

Open Source versus Proprietary Code Evaluation

Open source software is the development and maintenance of code in a shared code repository that collects work from many contributors and that enables anyone to read and test the code[29]. For a long time, advocates have claimed advantages to open source programing because the code can be reviewed by many independent people, thereby exposing bugs far earlier than catching them after deployment. A smaller number of people have advocated open source for software in the cybersecurity field because a similar argument applies: if anyone can read the code, a “white-hat” analyst will find any “zero-day” defect before a “black-hat” can detect the defect in the deployed system and then exploit the defect.

It is easy to agree with the principle of open source software: if the code is well reviewed, it is trustworthy. On the other hand, the open source principle should be backed up with empirical studies and these are lacking. Moreover, the original thrust of open source was the development of working, bug-free code. The security defects don’t quite fall in the category of a bug. Software may work flawlessly in its intended application and yet contain a fatal flaw when exposed to an unanticipated attack. Finally, there is the problem that there are not enough people with the talent and free time to check all the code[30]. It seems fair to say that open source has not been shown to work for the kind of software needed for cybersecurity.

Proprietary Code Evaluation. Clearly, the majority of software vendors will be unwilling to open the source code to their products because that action would provide too much access for the vendor’s competitors. On the other hand, the customers — particularly large corporations and governments — have a strong case that they must evaluate a vendor’s product for congenital and inflicted security defects. A compromise is possible because the customers are not competitors to the vendor. In the

[29] See for example: https://opensource.com/resources/what-open-source

[30] *A longer argument is given in our blog:* *http://ectn.typepad.com/pygar/2015/07/cyber-bugs-run-deep.html*

compromise scenario, the vendor establishes an evaluation center where the customer has full access to equipment and code. Although the customer would not be permitted to copy code, the customer's staff can review and test the code at the evaluation facility. In addition, the customer should be permitted to derive digital signatures for all the software components that will be installed at a customer's site. If the customer has a certified digital signature for the software components, it is technically possible to verify the delivered product by deriving the signature of the installed software. Such signatures protect the customer from alterations to the code after the review.

We call this compromise "proprietary code evaluation" because it maintains the competitive advantage of the vendor while allowing the customer the access required to detect and avoid congenital and acquired software defects. The disadvantages are the cost to the vendor for creating a secure evaluation center and the cost to the customer of providing staff to conduct the evaluation.

This approach to code evaluation has been implemented at least once. When the Chinese firm Huawei was entering the market with Internet routers and gateways in competition with western incumbents such as Cisco, the British government raised the concern that any software contained in the Huawei hardware might incorporate security flaws that were intentionally inserted. One might raise the same concern about software from the incumbent vendors given their cooperation with the NSA's internet monitoring operations. But, fairly or unfairly, the concern was raised in the case of Huawei. In response, a special evaluation site opened in Banbury, Oxfordshire, UK near the headquarters of GCHQ. This operation was initially reported in local press and *The Economist*[31] and updates were released in 2015[32]. The facility was given the name *Huawei Cyber Security Evaluation Centre* (HCSEC). It is impossible to ascertain the technical details of the HCSEC's implementation. However, the official judgement is that it is successful:

> *GCHQ has advised the Oversight Board that it is confident that HCSEC is providing the technical assurance of sufficient scope and quality as to be appropriate for the current stage in the assurance framework around Huawei in the UK.* [33]

[31] http://www.economist.com/node/21559929

[32] http://www.techworld.com/news/security/no-evidence-that-huawei-is-spying-on-uk-say-gchq-spooks-3605828/

[33] *For the 2015 Report of the HSEC see: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/416878/HCSEC_Report.pdf*

Code Signing – Single and Multiple Signatures

It is fairly standard practice today to insist that the software running on a system must be signed by a known, trusted party, who is the source of the software. Typically, the trusted party advertises that they have conducted exhaustive testing to be sure that the software is free of defects or security flaws. The signature only verifies that the software module that is received by an installation has not been modified after it was signed by the trusted source. If the source is mistaken, then any defect in the software will be installed undetected. In addition to this limitation, there are two other criticisms to be made about the code signing process.

First, in many cases, the software source is completely unknown to the organization installing the software. Should the installer trust the source or not? The answer in current practice is that the software source may sign the code using a certificate that has been signed by another organization that is known and trusted by the installer. To express this differently, the installer trusts a one organization, say Microsoft, but does not know enough to trust a smaller vendor, e.g. Logitech, that is the source for a necessary device driver. However, the smaller firm will ask the larger firm to certify its certificate. This step creates a chain of trust from the small vendor to the larger, well known vendor. In most cases, the software device driver will be installed based on that traceable lineage of certificates. The weakness of this common procedure is that the larger, trusted firm is too busy to watch everything that happens in the small firms for whom it issues a certificate.

If you have read this far in this book, you will realize that the chain of trust certificates creates an ever-expanding roster of insiders working at firms large and small. Each of those insiders has the potential to introduce a security flaw in a piece of software and sign the code as trusted. There is no practical way for the large firm that issues the well-known certificate to go around and monitor the insider threat at all their small-firm suppliers to ensure that no malignant code gets signed. Indeed, there are reports that that just this scenario happened during the STUXNET attack against Iran, although such reports are impossible to verify. Reportedly, an update to driver software written in a small Taiwanese firm was modified to contain a virus, then properly code signed, and then delivered only to Iranian users who introduced the virus to their systems by accepting an update from “Microsoft”.

Second, it is possible to argue that the source of the software is exactly the wrong group to review it for flaws. In any other branch of engineering, a safety critical design would be reviewed by an independent verification and validation firm that is entirely separate from the creators of the design. Software poses risks as well and ought to be independently verified. If that takes place, then the source firm and the verification (IV&V) firm both apply signatures to the same code artifact. Code that has multiple signatures is, if you can find it, less likely to contain a software defect.

In summary, we should point out that this discussion has drifted past the topic of what you should know and started to talk about what you ought to be asking for from vendors. The justification is that you should really know about code signatures. Next, we cross the line completely to talk about white-lists, which are not standard practice.

White-Listing

If it makes sense to check any software module that is added to a system for its code signatures, it ought to make sense to verify that all the modules in a system were added by this careful process. Are there any modules in the system that were not added as part of a secure, controlled, update process? It would be simple to check if installations maintained a list of authorized, signed modules. That list would be the white-list: a list of software authorized to be present.

In practice it is not easy to create a white list because the computer arrives with installed software and the vendors don't maintain a list of what software is delivered.

Similar Approaches (to be written)

The idea of processing encrypted information is an attractive solution and many people have investigated it. In this section, we describe other investigations. Each fills a need. However, all these other ways are predicated on a security risk from administrators within the central information repository. Unlike the Third Way, each of the other approaches discounts the insider threat from the many authorized users working for the client organizations that participate in the central repository. Nevertheless, the projects represent useful, instructive steps towards the Third Way.

Callisto (to be written or redirected)

Discussed earlier. See section on Callisto page 46.

IBM – Anonymous Resolution

Derives links to encrypted data only. Limited security of session key and no checkpoints or bailout points. Designed for gambling and applied to counterterrorism. Offers a single purpose discovery method and always returns links rather than data. Limited support for generalization and extension. Productized. IBM Anonymous Resolution – this product is no longer supported and IBM has withdrawn the documentation. However, press releases remain available, e.g. https://www.computerweekly.com/news/2240061215/IBM-weaves-cloak-of-anonymity-for-data-sharers

MIT – CryptDB

Very little protection from insiders (except data base administrators). The project was highly instructive because the encrypted processing is general and extensible via SQL. SQL proved very successful here. Continued development is unknown.

http://css.csail.mit.edu/cryptdb

http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb.pdf

Commentary on the Theory (to be written)

The author's purpose for this book has been to provide practical, technical guidance on the subject. Along the way it has been tempting to digress and explain how cooperation emerges under the right conditions and how the Third Way creates such conditions for Internet cooperation. In this supplemental chapter, we offer some comments on these topics of social organization and behavior.

Agency Theory (to be written)

Evolution of Cooperation (to be written)

Relationship with Game Theory (to be written)

Discuss evolution of cooperation and Law of Requisite Variety: competition, unit of selection.

Work for the Future (to be written)

Adjudication of Disputes (to be written)

Perhaps rewrite the description starting on page 37 of an older Conceptual Architecture description.[34].

[34] https://www.wwnsoftware.com/pdf/architectureV3.pdf

Index

Table of Figures

Notes

[i] Photo references: Alta Badia - Trentino Alto Adige, Italy, Guisepe Milo https://www.flickr.com/photos/giuseppemilo/

[ii]**Matching DNA and Protein Sequence**. To understand the matching operation, it is necessary to know that both DNA and protein molecules are polymers, that is, long chains of short molecules. The units of the polymers are nucleic acids or bases for the DNA and amino acids for the proteins. A gene sequence is stored in a data base as a sequence of bases. Living systems use only four of the possible bases. When a protein is produced according to the formula in the gene sequence, three bases are read together to determine which amino acid will be added next to the growing protein molecule. Because each of the three bases can have four possible values, a three base sequence or codon has 64 possible values. The codon value is then translated to a nucleic acid by use of a coding table implemented by tRNA molecules. The coding table matches the 64 codon values to a smaller set of 20 standard nucleic acids plus there are three stop codons that mark the end of sequence. The matching problem is now clear. The codon specifying the nucleic acid in the protein may have multiple values. The matching procedure needs to account for this ambiguity.

A secure matching mechanism within the context of the third-way has yet to be developed and may require an additional fiduciary to maintain privacy of the data and query while the two are matched.

Finally, it may be of interest that we humans use 22 not 20 amino acids to specify the monomers in a protein. The two extra amino acids are specified in the gene sequence by preceding a stop codon with a special codon sequence. Computer engineers will recognize the similarity of this mechanism to the escape codes used to add protocol and control-key codes to the limited 7-bit ASCII character code table.

[iii] **Encryption of Questions and Searches**. The search patterns, matching algorithms, and queries are all encrypted for two important reasons. First, a query can be just as sensitive as the data it asks about. For example, asking about a potential criminal could show that the person is under investigation and allow them time to flee. Second, encrypted comparison and search methods require the use of the same encryption key for both the information sought and the pattern which seeks information.

[iv] **Identity Authorities**. It seems worth noting several points. First, while there are some successful examples of an identity authority, many populations are not served by them. Second, it may require enabling legislation to improve the situation. Finally, the identity authority faces some difficult use cases.

The most common identity fiduciary that you will encounter is the *Certificate Authority*. A certificate authority uses *Public Key Encryption* to verify identity. The system that receives the certificate containing a unique private key can use that certificate for identification purposes. It is the responsibility of the certificate authority to ensure the actual, real-world identity of the certificate holder. The fiduciary responsibility of the authority rests in its application of careful identity verification procedures. We can cite, for example, Symantec and GoDaddy as two of the market leaders for commercial identity certificates. When you see one of these certificates you know that the certificate authority has done some research to establish the identity. Typically, the certificate authority will verify your business reputation with a service like Dun & Bradstreet and consider how long you have operated a business under that identity. On the other hand, they might do no more than call the phone number on the application to verify that someone answers. Generally speaking, you just don't know how well the certificate authority has exercised its fiduciary responsibility.

When a certificate authority like Symantec checks an identity with another authority like Dun & Bradstreet you can probably trust the certificate for a business. A certificate from a private citizen, on the other hand, can be based on no more than a cursory reference check. Private citizens are not well served by the current certificate authorities.

When MIT had a need for identity management to support campus-side distributed computing, they developed and deployed the Kerberos system. If a student, faculty member, or researcher had an identity in the MIT administrative system, that person could get a digital identity through Kerberos. Kerberos worked well because the extent of the "society of distributed systems" was limited to the campus. The use of central administration enabled a student to have a desktop computer that worked in the distributed environment. Although Kerberos is widely used elsewhere it remains unclear how it can organize an ad hoc on-line society without outside assistance.

The benefits of the third-way will be lost if the participants include imposters or transient identities with no commitments. For that reason, we've claimed that identity authorities are essential to the third-way. How might the utility of those authorities be improved? If there existed interagency cooperation in the Federal Government, the government might serve the valuable role of identity certification in the digital world. Even today, the Internal Revenue Service (IRS) is authorized to create identities for taxpayers so that everyone's income, obligations, and tax payments are properly assigned. That is turning out to be hard for the IRS as a result many people are scamming the system by claiming refunds for different people. On the other hand, the federal government originally created - but no longer owns - the US Postal Service which can easily deliver a certified mailing to any physical residence. That capability could be used to establish a connection between digital identities and people living at addresses. Undoubtedly, it would require new enabling legislation to allow the IRS or USPS to operate an identity management system but the public would benefit from putting an end to uncertainty over identity. Removing that uncertainty would make identity theft very difficult if the IRS simply enforced a rule that each person uses one address as a principle residence.

Finally, let us mention two difficult use cases for any identity authority:

- Assume the owner of a certificate loses the certificate and asks for a new one. If the certificate authority keeps a copy, they can provide a duplicate. That, however, implies that the certificate can be stolen from two places: the owner and the authority. That is not ideal.
- Assume that a certificate is stolen, how can anyone prevent a third party from using the certificate to pose as the owner and commit a crime? Worse yet, what if the certificate authority keeps the private key as a backup service for clients and its long list of keys is stolen? This actually happened in 2011 to RSA (see http://www.darkreading.com/attacks-and-breaches/rsa-securid-breach-cost-$66-million/d/d-id/1099232?)

[v] Dedication of the Essays: The documents used to demonstrate encrypted matching were written in the 17th century and have long since passed into the public domain. Nevertheless, authors and their funding agencies should have some rights and privileges in perpetuity; therefore, I feel it fitting to copy the dedication of the documents as written by Francis Bacon to afford him a voice in the present day and to acknowledge his source of research funding.

The Right Honorable
My Very Good Lord
the Duke of Buckingham
His Grace, Lord
High Admiral of England
Excellent Lord:

SALOMON saies; A good Name is as a precious oyntment; And I assure my selfe, such wil your Graces Name bee, with Posteritie. For your Fortune, and Merit both, have been Eminent. And you have planted Things, that are like to last. I doe now publish my Essayes; which, of all my other workes, have beene most Currant: For that, as it seemes, they come home, to Mens Businesse, and Bosomes. I have enlarged them, both in Number, and Weight; So that they are indeed a New Worke. I thought it therefore agreeable, to my Affection, and Obligation to your Grace, to prefix your Name before them, both in English, and in Latine. For I doe conceive, that the Latine Volume of them, (being in the Universall Language) may last, as long as Bookes last. My Instauration, I dedicated to the King: My Historie of Henry the Seventh, (which I have now also translated into Latine) and *my Portions of Naturall History, to the Prince: And these I dedicate to your Grace; Being of the best Fruits, that by the good Encrease, which God gives to my Pen and Labours, I could yeeld. God leade your Grace by the Hand. Your Graces most Obliged and faithfull Servant,*

Fr. Sr. Alban

The modern reader will perhaps note that Bacon proposes an alternative to XML coding to preserve information for future systems. Latin is pretty much out of the picture now but the jury is still out for XML.

[vi] When floating point parameters have been concealed with an order preserving encryption, the market maker can identify an overlap. When the market maker needs a specific number like a proposed *cash_per_share* it can use some reasonable procedure to pick the number from the overlap range. For example, one might take the midpoint of the encrypted overlap range. The encrypted midpoint will not be in the middle of the unencrypted range but it is not an unreasonable compromise number for the agreement between two players.