2008

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

MS CSIS Capstone Project

Cipher-N — a Secure, Web-based Document Exchange Service

Steven B. Sutton

12/8/2008

This report details the design, analysis, and implementation of a Web-based, hosted document exchange service called Cipher-N. Further, the report summarizes the experiences of the author as they pertain to the Cipher-N project as a basis for a Capstone experience.

Table of Contents

1 Introduction

Everyday individuals and businesses exchange sensitive electronic data such as PDF, Word and Excel files, with clients, customers or business partners. Unfortunately, a large percentage of these transactions are performed using unprotected methods of exchange such as email or FTP. Intercepted communications, compromised servers, or mismanaged email—all can result in the loss or unnecessary exposure of valuable proprietary and customer information. Organizations experiencing such an event can suffer the loss of existing customer confidence, damage to their brand, loss of a competitive advantage, and loss of future revenue from new customers that take their business elsewhere. Thus, in order to survive and prosper, organizations must always strive to find more effective means of protecting their data without compromising their productivity.

One way to protect data is through the use of encryption. Encryption is the process of transforming information using an algorithm to make it unreadable to anyone except those possessing special knowledge. Historically, however, solutions utilizing encryption to protect the exchange of documents have been costly and/or difficult to use. Hence, market opportunities exist for a secure, cost-effective, and user friendly solution.

The Cipher-N project is based a concept developed by Charles Laymon of StepQuest, Inc. The fundamental goal of the CapStone project was to develop a proof of concept utilizing Silverlight 2 as centerpiece of the development effort. The analysis, documentation, and software framework developed definitively show that the concept is feasible and it forms the basis of a future commercial offering

In the sections that follow, I discuss an approach to solving this problem (a Web-based, hosted document exchange service) and how I used it as the basis for a capstone project. I discuss the work completed thus far as well as aspects of the project that were analyzed and documented, but not completed as part of my capstone experience.

The paper is broken into four majors sections: business case, software architecture and design, scope, and lessons learned. The business case section details who is targeted with the solution, competing products, and what differentiates this solution from others. In the software architecture and design section, the justification for the chosen technologies and methodologies are discussed, the project timeline is outlined, and concrete implementation details are provided. The final two sections address exactly what was delivered and what was learned from doing so.

2 Business Case

2.1 Service Description

Cipher-N is a project designed to provide its clients with a user friendly, highly secure means of exchanging sensitive data with friends, family and business partners. In essence, Cipher-N is a Web-based, hosted document exchange service, which provides subscribers with the toolset necessary to securely exchange data without the need to fully understand the underlying security technologies.

Users of Cipher-N create an account on the website and identify partners for exchanging data. Using the browser-based interface, recipients of the data are selected and documents to be sent are locally encrypted prior to being uploaded over a secure SSL tunnel. Once on the server, the data remains encrypted and recipients are notified of its availability. To retrieve the data, recipients logon to the website and initiate a retrieval request. The data is downloaded over a secure SSL tunnel and decrypted on the client. Once retrieved by all intended recipients, the data and encryption keys are destroyed.

2.2 Target Market

Many of today's households and businesses are interconnected via the Internet. A number of these households and businesses have a need to exchange sensitive electronic data such as PDF, Word and Excel files, with clients, customers or business partners. Moreover, that need will grow as we move further toward an electronic information-driven economy. Presently, however, a large percentage of these transactions are performed using insecure forms of exchange. Of the estimated multi-billion e-mails sent daily only a small percentage are encrypted; leaving the remainder and their data highly susceptible to interception. Fortunately, businesses and consumers are becoming more aware of these risks as evidenced by the $61 billion dollars spent on IT security in 2006 by US companies. Of that total, $27 billion is spent on acquisition of new security technologies and services (1).

Cipher-N will target the security conscious consumer by providing a secure, cost-effective method of exchanging data with clients, customers or business partners. Cipher-N's target markets are individuals, private organizations and small businesses with a need to conveniently and securely exchange electronic data. These groups were chosen because they are most likely to have a need to transmit data considered sensitive, and unlikely to have the in-house technical expertise or desire to maintain the infrastructure necessary to effectively accomplish the task. Within these market segments, we are paying particular attention to the non-technical consumer who lacks the

necessary background to utilize online security tools currently available. Example customers include:

- Real estate agencies
- Law firms
- Affinity groups
- Financial service providers
- Security-minded individuals

As an example, consider the document exchanges required for securing a mortgage, which may today transpire without ever physically entering the lender's office. In part, the process requires submission of documents such as pay stubs, W-2s and bank statements, containing detailed financial about the borrower. In many instances, these documents are transmitted by unprotected e-mail, each time representing an opportunity for interception. Transmitting the documents via the secure Cipher-N service eliminates that opportunity, since the information is protected by end-to-end encryption.

2.3 Value Proposition

Based on Cipher-N's innovative approach and multi-platform technology, Cipher-N will bring significant value to users in the legal, financial, and real estate market segments. Cipher-N aims to enable the non-technical personnel to perform highly secure transactions. By accomplishing this, the solution offers tremendous value to all end users. These values include:

Ease of Use

An intuitive, graphical user interface allows minimally trained personnel to securely send documents in a secured manner with no need to understand the technical details of security mechanisms. Transmitting files in a comparatively secure manner is possible, but requires technical expertise and more effort. Users can focus on their primary business, rather than on technical expertise and infrastructure to secure their data communications.

Competitive Advantage

Cipher-N allows businesses to build a trusted brand and differentiate themselves by acknowledging their sensitivity and awareness of data privacy issues. Consumer concern for data privacy is at an all time high as evident by recent legislation activity around the country. As of July 18, 2006, at least 34 states in the U.S. have passed laws requiring organizations and government

agencies to notify customers, employees, and other affected individuals when a breach of protected personal information occurs due to human error, technology problems, or malicious acts (2).

Compliance

Utilizing Cipher-N, individuals and businesses have a secure and readily available audit trail for all of their document exchange activities. This capability helps to eliminate uncertainness in who has received a copy of the materials and keeps the sender better informed. In addition, larger, publicly traded businesses can utilize the audit trail capabilities to facilitate compliance with the reporting requirements mandated by the 2002 Sarbanes-Oxley Act.

Reduced Liability

Cipher-N's highly secure method of exchanging documents reduces the risk of exposing valuable proprietary or sensitive customer information. Leaks of sensitive customer information and other corporate data result in substantial costs (in terms of dollars and reputation) to the affected business. According to a recent study, in 2006, US companies lost on average $182 dollars for every compromised record due to direct increment costs, lost productivity, and lost customer opportunity (2). Thus, the high cost associated with compromised data makes a strong case for more strategic investments in preventative measures such as encryption.

Recipient Management

Cipher-N's message status capabilities enable a sender to quickly ascertain the list of recipient and their current status; thus, eliminating the need to follow-up with phone calls, e-mail, or track package delivery. Automatic receipt notifications let senders know when the data was received and by which recipient.

Reduced Operational Costs

When compared to physical transport and electronic alternatives, this service requires less up-front investment and less per-message cost. No pre-staging of documents for shipment, no manual shipping label preparation, no need to hand-deliver or arrange for courier shipment, and no per-recipient delivery fee. No technical infrastructure needs to be established or maintained.

Improved Productivity

The Cipher-N service allows users to manage the delivery of documents to groups of recipients. Delivery is tracked to all parties. No follow-up by phone is necessary; a quick glance at the Cipher-N report will tell senders if the message has been received by all recipients.

2.4 Features

	Feature	*Benefit*
Sender	Web-based	A browser-based interface allows sending documents from virtually anywhere.
	2,048-bit encryption	Industrial strength cryptography ensures your customers data is protect and your brand remains trusted.
	Recipient management	Real-time message tracking capabilities means you're always up to date.
	Audit trail	Message tracking capabilities ensure your always in compliance.
	User friendly interface	Enjoy an easy to manage and east to understand Windows style interface.
	Cost Effective	Quickly and affordably send documents to multiple recipients with a few keystrokes.
Recipient		
	Web-based	A browser-based interface allows document retrieval from virtually anywhere.
	2,048-bit encryption	Your data is always protected.

Table 2-1 Cipher-N Features

2.5 Competitive Assessment

The competition for document exchange comes from a wide variety of other software products and vendors. Several large web-based mail providers have means of transmitting documents, but they are not considered direct competitors, since they do not provide end-to-end protection of the data. Additionally, the niche market for highly secure exchanging of documents is rather un-crowded; leaving what an underserved market with opportunity for improvement.

Below I examine products and vendors that compete directly or indirectly in the target market segment.

Hushmail

Hushmail (http://www.hushmail.com) is a Web-based e-mail service that lets you send and receive email securely. Hushmail messages, and their attachments, are encrypted using Open PGP standard algorithms. These algorithms combined with Hushmail's unique key management system offer users secure message delivery. Hushmail's security is end-to-end; messages are encrypted before leaving the sender's computer and remain encrypted until after they arrive on the recipient's machine, where the contents are automatically decrypted (3).

Service Offerings

Hushmail's Web-based e-mail services are grouped into three distinct packages: standard, premium, and business

Standard

- Spam filtering and virus scanning to keep your Inbox clean
- File storage and sharing with other Hushmail users
- Unlimited contacts
- External POP3 access to other email accounts
- Hushmail Express for easy encrypted communication with contacts at any email address
- Hush Messenger for secure instant messaging
- Email notification
- Read receipts, auto-responders, drafts, and templates
- Extensive help resources
- Digital signatures for email and attachments
- End-to-end encryption for email and files
- 2,048-bit encryption with full OpenPGP support

Premium

- Desktop access to send and receive mail with Outlook, Thunderbird, and more
- Hush Secure Forms to receive encrypted email from your website
- 250 MB storage for email and secure file storage
- Large attachments up to 25 MB
- Unlimited pseudonyms to protect your email identity
- Technical support for quick answers
- No advertisements
- No deactivation due to inactivity

Business

- Secure email at either your own domain, or your-domain.hush.com
- Unlimited accounts with one monthly bill
- Domain administration to customize and manage your email domain
- Email archiving and policy management tools

Target Market

Hushmail competes in the horizontal market space of ad-supported and for-pay Web-based email that is currently dominated by Yahoo and Live Hotmail. There are no official statistics comparing the user numbers of the different web-based email address providers. However, techcrunch.com estimates that, of the approximately 500 million web-based accounts, greater than 95% are provided by Yahoo and Live Hotmail (4).

Hushmail differentiates itself within the Web-based e-mail market segment by focusing heavily on the security aspects of it product. Within the vertical market segment of secure Web-based e-mail, Hushmail is a very strong competitor. Of the competition, Hushmail is the only provider offering end-to-end encrypted e-mail and file storage and sharing services. In addition, facilities for publishing secure web-based forms (used to submit encrypted information from website visitors directly to Hushmail accounts) are provided.

Product Comparison

Criteria\Product	*Cipher-N*	*Hushmail*
Security	Very Secure	Very Secure
Ease of Use	Good	Good
Platforms	Windows, Linux, Mac OS X	Windows, Linux, Mac OS X
Technologies	Silverlight, .Net	Applet, Java
Source Model	Proprietary	Open
Email Based	No	Yes
Browser Plug-in Required	Yes	Yes

Table 2-2 Product Comparison

Alternative Solutions

Courier

Description: A courier is a person or company employed to deliver messages, packages and mail.

In a typical usage scenario, paper copies of documents or some other medium for storing digital data is packaged for physical shipment by a contracted courier.

Pros

- Secure

Cons

- Slow
- Expensive

File Transfer Protocol

Description: The File Transfer Protocol (FTP) is a protocol, standardized by the Internet Engineering Task Force (IETF), for exchanging files over TCP/IP based networks such as the Internet (5). The main goal of the protocol is to promote the transfer of files between computers while shielding the user from file system variations among different hosts.

In a typical usage scenario, servers which implement the protocol listen for client connection requests. Once connected, the client, which also understands the protocol, has available various file manipulation commands, including commands for uploading and downloading files to and from the server.

Pros

- Open standard
- Large number of free clients available
- Ubiquitous server support for the protocol

Cons

- Insecure – passwords and file contents are unprotected in transit
- Requires sender or receiver to maintain infrastructure (costly)
- Technical barriers (costly)

Application-specific Password Protection

Description: Most of the common office applications in use today (e.g., Microsoft Office, Adobe PDFMaker, and Sun StarOffice) offer some form of native password protection.

In a typical usage scenario, the document's author selects the protection option within the application and enters a password. The application is responsible for encrypting the document contents and embedding the password. To open the document, recipients must enter the same password, thus allowing the application to decrypt the contents for viewing.

Pros

- Easy to use
- Built into the application

Cons

- Application-specific
- Insecure - weak cryptographic algorithms
- Requires secure method of password exchange
- No recipient management

Electronic Mail

Description: Internet electronic mail (e-mail) refers to a store-and-forward system of world-wide electronic communication systems for sending and receiving messages based on the Simple Mail Transfer Protocol (SMTP) (6). In addition to simple text, email message can be used to transmit non-text attachments such as office documents.

In a typical usage scenario, users create or collect various types of data files on their local PC. Using an email client, these documents are then attached to an e-mail message for distribution to one or more recipients. Using an email client, on what can be a different host platform; the recipient retrieves the email message and optionally extracts the attached documents for reading or processing.

Pros

- Open standard
- Relatively easy to use
- Built into the application
- Large number of free clients available

Cons

- Insecure – passwords and message contents are unprotected in transit

Pretty Good Privacy and GNU Privacy Guard

Description: Pretty Good Privacy (PGP), and its open source replacement GNU Privacy Guard (GnuPG), refer to cryptographic software originally developed by Phillip Zimmerman in 1991 (7). Among other things, the software can be used to encrypt data, such as office documents, prior to exchange using asymmetric (public key) cryptography.

In a typical usage scenario, users each create a public/private key pair. The resulting public keys are then exchanged with other users via e-mail, Internet key servers or some other means of transport. This allows e-mail client plug-ins, which automate key management and cryptographic processes, to utilize GnuPG and recipient public keys for sending encrypted e-mail. Encryption of documents outside the e-mail program is also possible.

Pros

- Open standard

- Relatively easy to use
- Built into the application
- Large number of free clients available

Cons

- Requires method of exchanging keys
- Client-based e-mail limits availability
- Individual digital certificates can be difficult to obtain
- Technical barriers

Windows SharePoint Services

Description: Windows SharePoint Services (WSS), or Windows SharePoint, is a free add-on to Windows Server 2003 made available by Microsoft, which offers basic web portal and intranet functionality (8). Of its many features is the ability to present authorization-based, version-controlled document storage.

In a typical usage scenario, document authors can publish data to the website and defined exactly who has the right to view it. Limiting access to the website to secure HTTP connections ensures sensitive data is protected in transit.

Pros

- Highly accessible
- Commercial, supported software
- Reasonable protection of data in transit

Cons

- Data is stored unencrypted on providers' server
- Requires sender to maintain infrastructure
- Overkill — more than necessary to achieve the goal

3 Planned Software Architecture and Design

3.1 Development Environment

Platform

Silverlight and the .NET Framework

The .NET framework is the name for a bundle of related programs from Microsoft which allows for developing and running programs written many available languages, such as C# and VB.NET, that produce libraries and programs conforming to the Common Language Infrastructure specification (9). The framework provides the base class library (BCL), the Common Language Runtime (CLR), and other components for running the applications. Included in the BCL are a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation.

Silverlight is a cross-browser, cross-platform implementation of the .NET Framework (known as the .NET Framework for Silverlight or CoreCLR) for building and rich interactive applications for the Web. It includes a subset of the Windows Presentation Foundation (WPF) technology which offers animation, vector graphics, and video playback capabilities. Silverlight UI markup of vector graphics and animations is accomplished using Extensible Application Markup Language (XAML) — Microsoft's declarative XML-based language used to initialize structured values and objects (10).

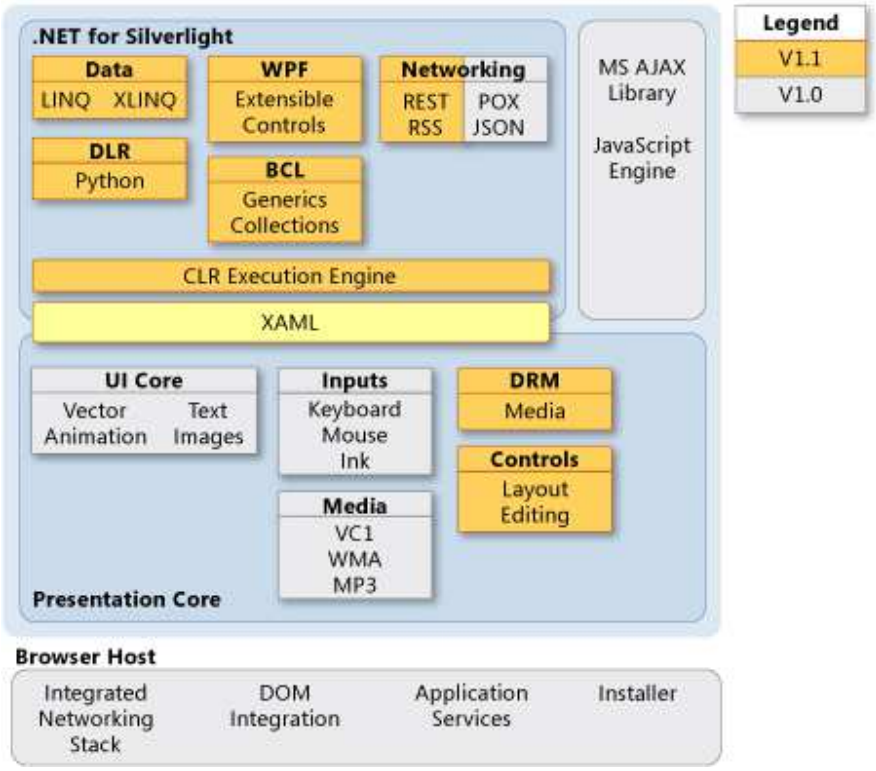

Figure 3.1 - Silverlight Architecture (10)

The Cipher-N project utilizes Silverlight and the .NET framework as its primary development platform. The framework was chosen for several reasons.

- The frameworks contains a wide breath of pre-coded solutions
- Silverlight is based on Windows Presentation Foundation
- C# is the author's preferred language
- Silverlight applications are cross platform

First, it contains a wide breadth of pre-coded solutions covering areas such as user interface, data access, web application development, and network communications. Second, Silverlight is a subset of Windows Presentation Foundation; hence, it provides the ability to build browser-based, portable, networked, media-rich Internet applications. Further, C# is the author's preference and a .NET language, allowing the same language to be utilized for client-side and server-side (e.g., database) development. Finally, Silverlight-based applications are cross-platform and run in most modern Web browsers; thus maximizing Cipher-N's base of potential customers.

Integrated Development Environment

Visual Studio 2008 Team Edition for Software Developers

Microsoft Visual Studio (VS) 2008 is an integrated development environment (IDE) which is a toolset for programming software applications (console and GUI applications including Windows Forms applications), web sites, web applications, and web services that run on any platforms supported by Microsoft's .NET Framework. Visual Studio also includes Silverlight-specific features, including IntelliSense, debugging, and Silverlight project templates that create and link all required files.

Visual Studio 2008 is the IDE chosen for Cipher-N development due to several factors. These include (a) support for end-to-end debugging capabilities for SQL Server hosted .NET assemblies and T-SQL stored procedures, (b) a width breadth of views, editors, wizards, builders, and code merging and refactoring tools, (c) tight integration with the chosen application lifecycle management tool (Team System), (d) Silverlight-specific features to facilitate development, and (e) the author's desire gain experience with Visual Studio 2008.

Relational Database Management System

SQL Server 2005

SQL Server 2005 was chosen as the relational database management system (RDBMS) for Cipher-N based on several criteria. These include (a) a low cost-to-user ratio, (b) ease of administration, (c) support in Visual Studio for end-to-end debugging of hosted .NET assemblies and T-SQL stored procedures, and (d) ability to manage unstructured data (e.g., bitmap images, text files) in the database while physically storing it on the file system.

Application Lifecycle Management

Visual Studio Team System 2008

Visual Studio Team System 2008 (Team System) is an application lifecycle management tool from Microsoft. Team System itself is comprised of several products, one of which is Team Foundation Server (TFS). TFS is a server-side component that provides a source control repository, work item tracking, bug tracking, and reporting services. At the work item level, TFS items can be distinguished by type, such as a Task, a Quality of Service Requirement, a

Scenario, and so forth. The client-side component of Team System consists of several versions of Visual Studio, called Team Editions, which are targeted to roles in the application development process.

Team System and the Visual Studio 2008 Team Edition for Software Developers was utilized by the Cipher-N project. The decision was based largely on the tight integration between the IDE and TFS and the author's desire to gain experience with Visual Studio Team System 2008.

Alternatives Considered

Flash

Flash refers to both the Adobe Flash Player (a browser-based client application), and to the Adobe Flash Professional integrated development environment. It supports vector and raster graphics, scripting via the ActionScript language, and bi-directional streaming of audio and video (11).

For the graphical user interface portion of Cipher-N, Flash was considered. Its strengths include a large installation base and small plug-in install size. However, Flash was not chosen due to several factors including the sole supported language is ActionScript and the limited (relative to the Java Platform and .NET Framework) set of class libraries.

Java Platform

The Java platform is the name for a bundle of related programs from Sun Microsystems which allows for developing and running programs written in the Java programming language. The platform provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in Java (12).

The Java platform was investigated for use in developing the graphical user interface and server-side components of the system. Specifically, the Java applet technology was considered due to its convenient browser-based deployment model and platform independence. However, the platform was not chosen due to the lack of tight integration with SQL Server 2005 and the author's desire to work with new technologies such as Windows Communication Foundation and Silverlight.

Eclipse

Eclipse is a Java-based open-source software framework that offers, among other things, a Java integrated development environment. The Java IDE component of Eclipse adds a Java project nature and Java perspective to the Eclipse Workbench as well as a number of views, editors, wizards, builders, and code merging and refactoring tools designed to maximize programmer productivity (13).

Eclipse was considered for use as the primary integrated development environment (IDE) for the project; however, it was not chosen for several reasons. Primarily, it does not presently support the C# and the .NET framework; hence, it lacks end-to-end debugging capabilities for SQL Server hosted .NET assemblies and T-SQL stored procedures. In addition, a language plug-in for the user interface markup language XAML (Extensible Application Markup Language) is not yet available.

Oracle

Use of Oracle 11g as the relational database management system (RDBMS) for Cipher-N was contemplated. However, due to its high cost-to-user ratio relative to other RDBMS products it was not considered a viable option. Further, usage of Oracle comes at the expense of several productivity features offered by Visual Studio 2008 when used with SQL Server including end-to-end debugging for stored procedures and .NET “TableAdapters”.

Trac

Trac is an open source, web-based project management and bug-tracking tool with an interface to Subversion (an open source version control system). The system was considered to provide application lifecycle management — the process of delivering software as a continuously repeating cycle of inter-related steps: definition, design, development, testing, deployment and management — for the Cipher-N project. Ultimately, Trac was not selected mainly due to the author’s desire to gain experience with Visual Studio Team System 2008.

3.2 System Development Lifecycle

Project Methodology

Unified Process

The Unified Process (UP) is an extensible, object-oriented software development methodology that is based on a few basic tenets of software development best practices: iterative and incremental development, use case driven functionality, architectural centricity, and risk focus. The development life cycles of projects following the UP are divided in to four distinct phases: inception, elaboration, construction, and transition, as illustrated in Figure 3.2 - UP Phases and Objectives. Phases are further sub-divided in to iterations—processes characterized by analysis, design, and implementation work activities—whose primary goal is an increment. Increments are system releases system that contains added or improved functionality compared with the previous release (14). Division into phases serves to define the serial sequence of the project each having a primary emphasis, or objective, on which the team focuses their activities. Project overhead is also reduced as subdividing eases the planning and management.

UP PHASE	OBJECTIVE
Inception	Develop an approximate vision of the system, make the business case, define the scope, and produce rough estimates for cost and schedule.
Elaboration	Refine the vision, identify and describe all requirements, finalize the scope, design and implement the core architecture and functions, resolve high risks, and produce realistic estimates for cost and schedule.
Construction	Iteratively implement the remaining lower-risk, predictable, and easier elements and prepare for deployment.
Transition	Complete the beta test and deployment so users have a working system and are ready to benefit as expected.

Figure 3.2 - UP Phases and Objectives (14)

Along with phases and iterations, the UP also defines a set of disciplines. A disciplines refers to a set of functionally related activates that together contribute to one aspect of the development project. Disciplines within the UP exist to make the process of iterative development more manageable. UP disciplines include business modeling, requirements, design, implementation, testing, deployment, configuration and change management, project management, and environment. Effort allocated to each discipline changes as the project progresses, as illustrated

in Figure 3.3 - UP Process Life Cycle. Naturally, early iterations focus or requirements gathering and modeling the system. Subsequent iterations shift the focus to implementation and testing with final iterations emphasizing deployment.

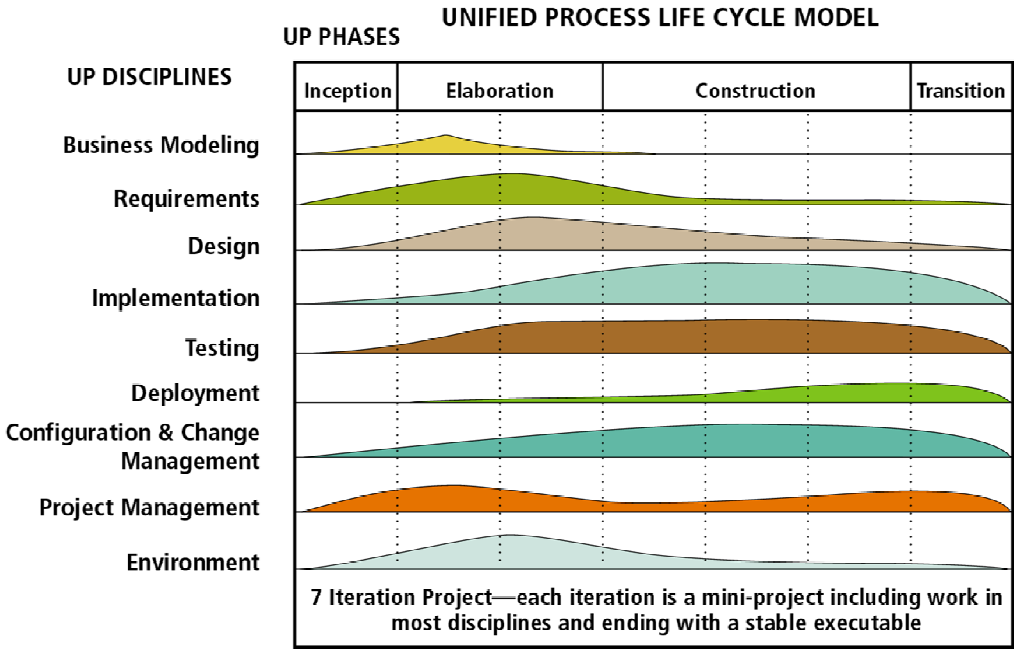

Figure 3.3 - UP Process Life Cycle (14)

The UP was selected for the project based on its risk-focused—indeed, several key aspects of the project present a high level of risk—and customizable characteristics. An initial development cycle consisting of five iterations (two in the elaboration phase) is currently planned.

Inception

The inception phase is one of the shorter phases in the UP and is largely characterized by project definition. Goals for this phase include (14):

- Development of project justification or business case
- Definition of project scope and boundary conditions
- Outline of use cases and key requirements that drive the design tradeoffs
- Outline of one or more candidate architectures
- Risk identification
- Preparation of a preliminary project schedule

During the inception phase, several data gathering techniques were utilized to achieve the stated goals. These included brainstorming, interviews, and external research.

Cipher-N is based on an idea developed by Charles Laymon of StepQuest, Inc. Brainstorming sessions with Charles Laymon and Dr. Douglas Kline of UNC-Wilmington were conducted on several occasions to flesh out the implementation details and drive towards a viable product offering. Key features of a potential product discussed during these sessions including focus on ease of use, end-to-end data encryption, and how to best convey an overall value proposition. Possible system architectures and user interface design were also sketched out during the “white boarding” sessions.

Several interviews with Charles Laymon were conducted to further elaborate on system mock-ups, produced initially by Charles Laymon and later expanded as part of the project, illustrating the concept (see Appendix A). Technical aspects of system architectures, such as data transport and deployment methods were explored as well as selection of cryptographic algorithms. In addition, risks and mitigation strategies associated with the use of cutting edge technology including alpha and beta cycle products were discussed. Project management issues were also visited including approaches to application life-cycle management, development tools and schedules.

Extensive external research was also conducted during this phase. Competitors, both companies and products, were identified and reviewed (see Competitive Assessment). Additionally, data collected during this phase is particularly important, since it helped to better identify features necessary to differentiate the product offering.

Elaboration

The second phase of the project (elaboration) is one of the longer phases in the UP and is largely characterized by detailed project definition. Goals for this phase include:

- Validation of architecture
- Definition of all system requirements
- Addressing high-risk areas
- Preparation of project schedule

The first iteration served to complete a large portion of the finalized system requirements. To do so, event decomposition techniques were utilized. Such techniques help identify use cases by

first focusing on the events a system needs to response to and then looks at how a system must respond (14). The list of events served as a starting point from which a detailed set of use cases were developed. The list of events— together with the trigger, source, use case, response and destination for the event (14)—identified for the Cipher-N system are summarized in Figure 1 – System Event Table of Appendix B. Associated use cases are also located in Appendix B. Figure 3.4 - Cipher-N Subsystems illustrates how each use case relates to the system as a whole; thus, helping to convey the breadth of requirements considered when developing the system.

Figure 3.4 - Cipher-N Subsystems

High-risk areas were also addressed in iteration one. Two such items were identified in the inception process: usage of Silverlight 1.1 alpha and availability of cryptographic algorithms in the CoreCLR.

The first risk, usage of Silverlight 1.1 alpha, stemmed largely from the inherent risk associated with using any software product in the alpha stage of development and the learning curve associated with a new product. It was addressed by a firsthand assessment of the available UI controls, supported framework namespaces, and overall system stability. Ultimately, it was concluded that the appropriate features were available and system stability was sufficient to proceed.

The second risk, availability of cryptographic algorithms in the CoreCLR, was a larger concern for a couple of reasons. First, the runtime associated with Silverlight is by design very small. Thus, its limited size limits the number of .NET Framework class libraries available to Silverlight applications. In the case of Cipher-N, several cryptographic algorithms are necessary to enable end-to-end data encryption. While several cryptographic algorithms are supported by the full .NET Framework, none are available in the CoreCLR. Further compounding the risk is the security model employed by the CoreCLR at runtime, which precludes the execution of unmanaged or “unsafe” code; thus, removing the ability to interact with encryption service offered by operating system APIs. To overcome this limitation, source code for the selected encryption routines was located and compiled into a Silverlight class library. Using the library's routines several tests involving encryption and decryption of files and text hashing were conducted. All testing results were satisfactory and the conclusion to proceed with Silverlight was made.

Detailed database modeling was also performed in this phase. A discussion of the database schema and design logic is located in the Section 3.3.

The remaining three iterations were completed following approval of the project proposal.

Alternatives Considered

Extreme Programming

Extreme Programming (XP) is an adaptive, agile development software development methodology characterized by iterations, spikes and testing. The development approach consists of three levels (a) system, (b) release, and (c) iteration (see Figure 3.5 - XP Methodology (14)). Each develop project cycle through the system level activities once, followed by multiple cycles of release and iteration. Measuring progress is a main theme throughout the XP process and check backs to the customer are frequent.

The methodology prescribes a set of stakeholder practices meant to encourage the “XP values” which are (a) communication, (b) simplicity, (c) feedback, and (d) courage (14). These practices include:

- Planning
- Testing
- Pair programming
- Simple designs
- Refactoring the code
- Owning the code collectively
- Continuous integration
- On-site customer
- System metaphor
- Small releases
- Forty-hour week and coding standards

Benefits

- System development more flexible with respect to changes
- Early and frequent nature of the tests helps to catch defects early in the development cycle
- XP creates working software faster than traditional approaches

Limitations

- Requirements instability
- Total lines of code are increased due to unit test implementation

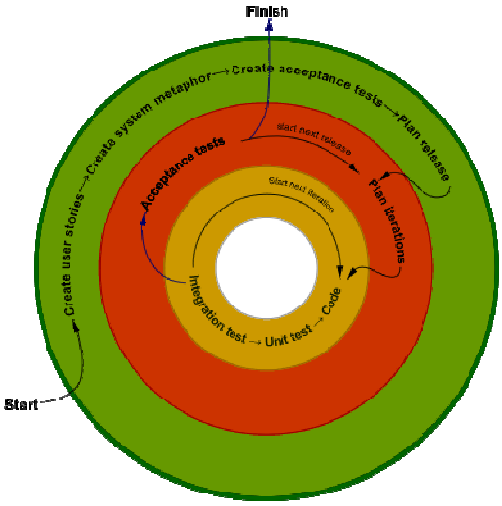

Figure 3.5 - XP Methodology (14)

Test Driven Development

Test Driven Development (TDD) is a software development methodology, which emphasizes testing prior to coding and a lineage that traces back to Extreme Programming (XP) (15). In essence, the iterative TDD process consists of writing a test for the new functionality you intend to develop, followed by implementing the code to pass the test, and finally refactoring (cleaning up) the source code. More concisely, TDD is defined by two simple rules: (a) Never write code unless you have a failing automated test; and (b) Eliminate duplication (16).

Benefits

- Leads to more modularized, flexible, and extensible code
- Early and frequent nature of the tests helps to catch defects early in the development cycle (16)
- Ensures 100 percent code coverage for software testing

Limitations

- Difficult to use for graphical user interface development
- Total lines of code are increased due to unit test implementation

Waterfall Approach

The waterfall approach is a highly predictive software development methodology that prescribes a set of sequential development phases: (a) requirement specification, (b) design specification, (c) implementation, (d) verification, and (e) maintenance (14). In this approach, each phase is discrete and, once completed and perfected, the phase cannot be revisited. This approach is illustrated in Figure 3.6 - Waterfall Model (14).

Benefits

- Simple, disciplined approach
- Strong emphasis placed on documentation
- Well understood requirements and design

Limitations

- Assumes perfection is achievable at every phase
- Not well suited for highly changing, dynamic environments with changing priorities on requirements
- Inability to revisit a phase once completed

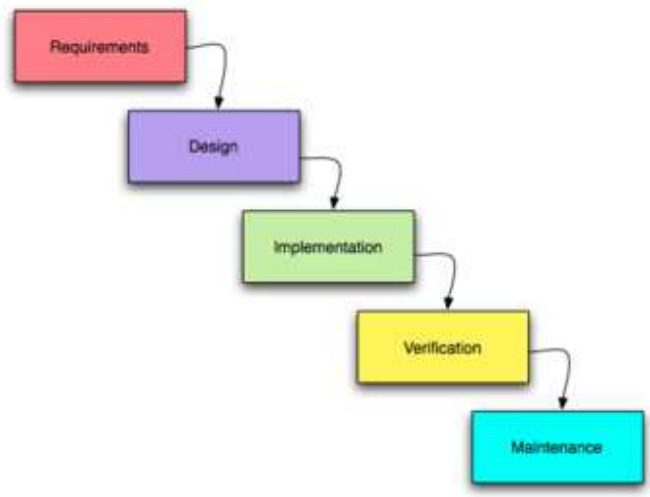

Figure 3.6 - Waterfall Model

3.3 Design

Cipher-N falls into a relatively new classes of applications known as Rich Internet Applications (RIA). RIAs are web applications akin to traditional desktop application in terms of features and

functionality. However, unlike traditional web-based applications that focus mainly on server-side code execution, RIAs transfer a significant amount of processing to the client. To do so, a client engine (Silverlight), along with any additional application code required, is downloaded to the client's browser at application startup. Subsequently, the engine accepts responsibility for application execution and associated activities such as rendering and network communication.

Architecturally, the system is fashioned after the well-known n-tier architecture, as illustrated in Figure 3.7 - System Architecture. In this architecture, the presentation layer, business logic layer, data access layer, and data layer are developed and maintained as independent modules—all with well-defined interfaces. Separating application components into separate layers increases the maintainability and scalability of the application. It does this by enabling easier adoption of new technologies that can be applied to a single tier without the requirement to redesign the whole solution (17).

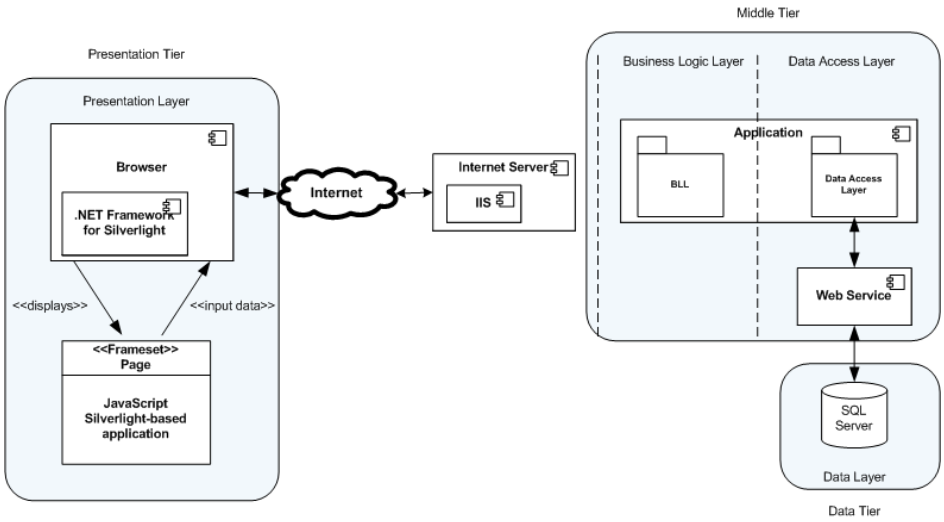

Figure 3.7 - System Architecture

Presentation Layer

The presentation layer of the project is embodied in a browser-based Silverlight application, through which all client interaction is conducted. Users of the service launch the application by browsing to a web page. Following authentication, users are presented with a wide variety of options including the ability to manage partners, update account information and exchange messages (and documents) with partners and non-partners (see Appendix B for use cases).

In the case of message exchange, users have two distinct options, which significantly impact the application behavior. The first is message exchange with a partner, which implies the recipient has either independently completed the registration process or done so following an invitation from a Cipher-N member. In either case, by registering with the system a public/private key pair unique to the partner is known by the system. The second type of message exchange involves an unregistered recipient or “non-partner”. In this case, a public/private key for the recipient is not known by the system and an alternate form of user authentication (i.e., secret question and answer) is utilized. Specifics of the encryption and decryption processes are illustrated through the use of sequence diagrams in Appendix C.

Business Logic Layer

The business logic layer (BLL) — the core of the application constructed according to the requirements — is also embodied in the browser-based Silverlight application. Within the BLL, the application implements the rules and procedures of business processing (14) and characterizes the semantics for presentation layer interaction. Businesses critical functionally, such as cryptographic algorithms, are also implemented in this layer. Another important aspect of the BLL is object representations of data. In .NET code, such representations come in the form of “DataSets”, “DataTables”, and object classes.

Data Access Layer

The data access layer (DAL) is responsible for handling the domain constraints on the data and for updating and retrieving the data from the data store. The system itself enforces the logical separation of the BLL and DAL through the use of the .NET “TableAdapter” construct and a Web service. In .NET, TableAdapters are responsible for performing insert, update, select, and delete operations against the database. BLL objects such as DataSets interact with the DAL through the use of their associated TableAdapter. The TableAdapters in turn call Web service methods exposed via IIS for specific data manipulation operations (Note: In current versions of Silverlight direct database communication is not supported).

Data Layer

Although composed of many more tables, the database is conceptually organized into eight entities or themes: access, account, contact point, domain, message, notification, offering, and party. The following describes the roles and responsibilities of each the logical grouping, after

which Figure 3.8 - Use Case Entity Matrix illustrates the actions performed on the entities as a result of each use case. A complete set of entity relationship diagrams are located in Appendix D.

Access

The central role of the Access entity is to organize data regarding authentication (any process by which you verify that party is who they claim they are) and authorization (finding out if the authenticated party is permitted to have the resource). Each party in the system is assigned one or more records in the Access table. Access table records are used in some instances to directly relate authentication credentials (e.g., username and password) to a party and in other instances to assign a party role membership. Access table records are further used in relationships that define the roles to which a party is ultimately assigned.

Account

The central role of the Account entity is to organize data regarding accounting information. Each access entity is associated with one or more methods of payment and a pay schedule. Details regarding the type of account, method of payment, and transaction information for payment instances are also stored.

Contact Point

The central role of the Contact Point entity is to organize data regarding points of contact for a party. These include phone numbers, addresses, email addresses and websites. Each party in the system can have multiple points of contact and multiple point of contact of the same type, which are further designated by subtypes such as work, home, mobile, etc. Records in the ContactPoint table serve as the focal point in relations linking a party to a contact point.

Domain

The domain entity serves as the epicenter of the Cipher-N database; hence, all data is directly or indirectly related to a Domain table record. A domain is a logical subdivision of the data. Organizing data in the domain hierarchy provides the flexibility to house data from multiple sites in the same schema, which could be leveraged to offering customer-branded hosting at some future point. The domain entity is also used to store top-level content associated with the domain.

Message

The central role of the Message entity is to organize data regarding the actual message sent by a party. Metadata such as the file type, encryption algorithms, and cryptographic keys are stored for a message along with the encrypted message contents. Additionally, user-definable constraints, definable at the party-level or message-level, are stored and related to the messages. Six constraints are defined:

Constraint Type	Description
Expiration date	Message expiration date
Pickup by date	Requested pickup date
Maximum retrievals	Maximum number of times a message can be retrieved
Use certificate	Certificate required for authentication
Use question and answer	Question and answer required for authentication
Use login	Account required for authentication

Notification

The central role of the Notification entity is to organize data regarding notifications generated and sent by Cipher-N. Notifications are triggered by events occurring in the system. The form and channel (or pathway) that such notifications are communicated are persisted by the tables in this entity, as is the status. For maximum flexibility, a notification is associated with a template (stored as ASCII, HTML, or XML), an event type and a channel (e.g., email, web service) for communicating the notification. Eight event types are defined:

Event Type	When Triggered
Welcome member	New Cipher-N member is registered
Invite partner	Non-member is added to a Cipher-N member's partner list
Reset password	User requests a password reset
Expired account	User's account expires
Message Ready	Cipher-N member submits a new message
Message Received	Recipient retrieves a message
Message Overdue	Message has not been retrieved by the scheduled date
Account Changed	Cipher-N member modifies their account

Offering

The central role of the Offering entity is to organize data regarding products offered for sale by Cipher-N and organized as a package. A package bundles products and services into cohesive customer offerings. Each package is composed of products and services, which are defined on a more granular basis by a set of features. Cipher-N will offer four package levels: basic, essential, business and enterprise. Features of products and services include, for example, encryption level, number of free support hours, maximum file upload size, total disk quota, and maximum number of partners.

Party

The central role of the Party entity is to organize data regarding Cipher-N users, such as individuals and organizations, and relationships among them. Each user of the system is assigned a record in the Party table. Party table records are used in some instances to relate a party with an organization and other instances to associate a user's partners. Party table records are further used in relationships that define a party's access and points of contact.

Use Case Database Entity Matrix

	Use Cases	Database Entities							
ID	Name	Access	ContactPoint	Domain	Party	Offering	Message	Notification	Account
1	Register new user	C	C		C			C	
2	Authenticate user	R		R					
3	Add new partner	CR	CR		C			C	
4	Query customer partners		R		R				
5	Encrypt new message	R	R		R	R	C	C	
6	View message status	R					R	R	
7	Upgrade account service level	U				R		C	
8	Update customer profile		U		U			C	
9	Lost account information	RU	R		R			C	
10	Merge accounts	RU	RU		RU		RUD	RUD	RUD
11	Cancel account (deactivate)	U					R	RC	
12	Decrypt message (partner)				R		RU	RC	
13	Decrypt message (non-partner)				R		RU	RC	
14	Produce summary report		R		R	R	RU	C	
15	Charge customer account		R					C	CR
16	Modify customer account	CRU	CRU		CRU			C	
17	Delete account	RD	RD		RD		RD	RD	RD

C=Creates new data
R=Reads existing data
U=Updates existing data
D=Deletes existing data

Figure 3.8 - Use Case Entity Matrix

4 Scope

The merit of the project is based on the successful analysis, documentation, development, testing, and implementation (limited) of the Cipher-N project. Sections 1-3 detail a complete specification for what is considered to be a complete version 1.0 implementation. However, the man-hours required to complete the project exceed the practical time available during a normal semester; thus, only a portion of the messaging subsystem illustrated in Figure 3.4 - Cipher-N Subsystems was targeted for completion. It was chosen for develop first, since it represents the crux of the product offering, whereas the remaining subsystems exist for augmentation and support.

At a minimum, support for the messaging subsystem required partial implementations of all of the system layers: presentation, business logic, and data access. Figures 6 and 12 of Appendix B illustrate the relevant use cases for the messaging subsystem. In essence, these use cases detail the end-to-end sending and receiving of an encrypted message and represent the scope of this project. The impact of each use case by application layer is summarized in Table 4-1 – Use Case Impact by Application Layer along with references to supporting artifacts.

Layer	***Component***	***Artifact***
Presentation		
	Silverlight UI	Appendix A - Figure 11 (Send Message) Appendix A - Figure 12 (Retrieve Message)
Business Logic		
	Encryption algorithms	Appendix C - Figure 1 (Encrypt Partner Message) Appendix C - Figure 2 (Decrypt Partner Message)
	Business entity models	
	Data adapters	
Data Access		
	Web service	
	Tables, stored procedures	Appendix D

Table 4-1 – Use Case Impact by Application Layer

As discussed in Section 3.3, work prior to the project proposal encompassed the inception phase and the first phase of elaboration. A timeline for the complete project (inception, elaboration, implementation, and transition) is detailed in Table 4-2 - Cipher-N Project Timeline, including man-hour estimates for each component of the system. For purposes of the Capstone work, the project is considered complete having successfully completed the following: implementation of the aforementioned use cases, submission of the written report, and completion of an oral defense as described in the UNCW MS CSIS Graduate Student Handbook.

	Task Name	Work	Start	Finish
1	**Inception Phase**	**74 hrs**	**Mon 7/2/07**	**Tue 8/21/07**
2	Brainstorm project idea	20 hrs	Mon 7/2/07	Fri 7/13/07
3	Team Foundation Server build-out	24 hrs	Mon 7/16/07	Tue 7/31/07
4	UI mock-ups	30 hrs	Wed 8/1/07	Tue 8/21/07
5	**Elaboration Phase - 1st iteration**	**100 hrs**	**Wed 8/22/07**	**Tue 10/30/07**
6	Database design	20 hrs	Wed 8/22/07	Tue 9/4/07
7	Research encryption algorithms	20 hrs	Wed 9/5/07	Tue 9/18/07
8	Spike Silverlight code	20 hrs	Wed 9/19/07	Tue 10/2/07
9	Use cases	20 hrs	Wed 10/3/07	Tue 10/16/07
10	Sequence diagrams	20 hrs	Wed 10/17/07	Tue 10/30/07
11	**Proposal**	**110 hrs**	**Mon 10/29/07**	**Fri 1/11/08**
12	**Elaboration Phase - 2nd iteration**	**40 hrs**	**Mon 1/14/08**	**Tue 2/5/08**
13	Spike Windows Communication Foundation	20 hrs	Mon 1/14/08	Thu 1/24/08
14	Class diagrams	20 hrs	Thu 1/24/08	Tue 2/5/08
15	**Construction Phase**	**205 hrs**	**Tue 2/5/08**	**Wed 10/22/08**
16	**Database scripts**	**55 hrs**	**Tue 2/5/08**	**Fri 3/7/08**
17	Loading	10 hrs	Tue 2/5/08	Mon 2/11/08
18	Structural	10 hrs	Mon 2/11/08	Fri 2/15/08
19	Cryptography	15 hrs	Mon 2/18/08	Tue 2/26/08
20	CRUD stored procedures	10 hrs	Tue 2/26/08	Mon 3/3/08
21	Custom stored procedures	10 hrs	Mon 3/3/08	Fri 3/7/08
22	**.NET coding**	**150 hrs**	**Fri 3/7/08**	**Wed 10/22/08**
23	UI - client	40 hrs	Fri 3/7/08	Fri 5/16/08
24	Server	40 hrs	Fri 5/16/08	Fri 6/20/08
25	Windows Communication Foundation	40 hrs	Fri 6/20/08	Fri 8/29/08
26	Data access layer	30 hrs	Fri 8/29/08	Wed 10/22/08
27	**Transition Phase**	**10 hrs**	**Wed 10/22/08**	**Tue 10/28/08**
28	**Capstone Defense**	**25 hrs**	**Mon 12/8/08**	**Mon 12/22/08**

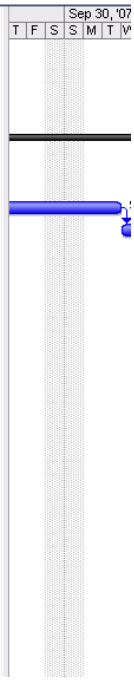

Table 4-2 - Cipher-N Project Timeline

5 Motivation

Motivation for my Capstone project derived from a variety of sources, but mainly from the desire to learn and contribute to the body of knowledge in areas of Computer Science and Information System. Specifically, the contributions I made focused on three distinct areas: contribution to the organization, personal professional development, and contribution to the profession.

5.1 Contribution to the Organization

The Cipher-N project is based a concept developed by Charles Laymon of StepQuest, Inc. All Capstone project deliverables (e.g. analysis, documentation, and software framework) and research was provided to StepQuest, Inc. for potential use in a commercial offering.

5.2 Personal Professional Development

In my current role as a software engineer, I am exposed to only a small subset of the .NET framework technologies dealing mainly with desktop applications. Aspects of the framework relating to Web applications and Web services are largely out of scope largely due to the highly secure nature of my business. As such, this Capstone project was selected, in part, to broaden my base of expertise and gain exposure to areas I would otherwise not typically work in. The proposed project achieves this goal, since it centers largely on Web-based technologies such as Silverlight and Windows Communication Foundation. Further, by working closely with a professional Web developer, I gained insight into Web development best practices.

5.3 Contribution to the Profession

New Technologies

The technology roadmap for the Cipher-N project includes several “cutting edge” technologies in various stages of the product life cycle. These include Silverlight 2.0 (formerly 1.1), Visual Studio 2008 and Visual Studio Team System 2008. As such, little data exists in the public domain regarding the merits of each in actual development scenarios. Utilization of the tools in my Capstone experience, affords me the opportunity to provide valuable feedback the development community including questions such as: In what situations do the new technologies make sense? What features helped achieve greater productivity?

Process / Methodology

Several software development methodologies were considered for use in the project. Ultimately, the Unified Process (UP) was selected for the project based on its risk-focused and customizable characteristics. However, the decision was made having no prior experience with the process, relying instead on literature review. Thus, utilizing the UP for development allowed me to gain a better understanding of how the process functions and ascertain whether it is suited for small scale, small team projects of this type. Further, has allowed me to investigate what tools in the UP (e.g., use cases, UML artifacts, and mock-ups) helped to best facilitate the project.

Database Design

As described in Section 3.3, the database design employed by the project employees a highly abstracted entity model. In addition, where possible, the use of CRUD stored procedures is preferred over a highly customized set specific to the application. These design decisions were made to build-in a high degree of flexibility for future application expansion. However, this degree of flexibility comes at the expense of a larger number of stored procedures (e.g. a minimum of four per table for CRUD operations) and the potential for higher number of stored procedure calls per transaction. Experience gained during the construction phase of the project provided a better understanding of the trade-offs for utilizing highly abstracted data models.

6 Project Review

Previous sections were authored prior to the project proposal in January 2008. This section reviews the project following completion and encompasses worked completed through December 2008. Sections 6.1 through 6.5 objectively state details of how the system was implemented, what processes were used, how it was tested, and systems metrics to date. Section 6.6 is a subjective analysis of the project.

6.1 Timeline

The original timeline (see section 4) called for completion of the project at the end of April 2008. Much work was completed at that time; however, work commitments of stakeholders, and issues with pre-release software products delayed progress.

Referring to the original timeline in Table 2 of section 4, .NET coding did begin as planned in March of 2008. However, the estimate of 50 hours for coding actually became approximately 150 hours. Breaking changes between Silverlight alpha and beta releases accounted for approximately 50 hours. Another 25 hours was devoted to rework as SQL Server 2008 moved from beta to release. Finally, refactoring efforts necessitated by exploration of cutting edge technologies accounted for the remaining 25 additional hours.

6.2 Implemented Software Architecture and Design

This section reviews the software architecture and design, specifically comparing the as-planned system to the as-built system. We present the system by tier: data, middle, and presentation. Architecturally, the final system was implemented largely as expected utilizing the well-known n-tier architecture. Figure 6.1 - Final System Architecture illustrates this design.

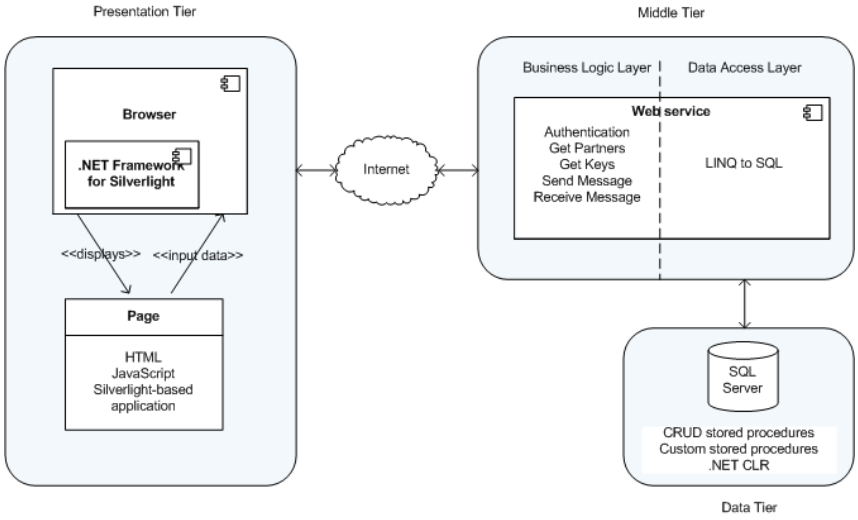

Figure 6.1 - Final System Architecture

A key design feature of Cipher-N architecture is the Silverlight control's direct interaction with the Web service. This ensures that connections for encrypted file transfer from the data vault are separate from connections for standard site Web pages. Although the ASP.NET Web site and Cipher-N Web service are currently deployed in the same IIS instance this is not a requirement. This allows for the complete separation of Web site hosting from Web service related activities. In the future, the encryption process may be hosted on a more physically secure server than the web server.

Data Access Tier

The data access tier was implemented through the use of stored procedures, the .NET CLR and a SQL Server 2008 database. The bulk of the stored procedures (SP) were basic create, read, update, and delete (CRUD) operations. The remaining SPs were higher-level custom stored procedures. Some CLR integration in SQL Server was also used.

The basic CRUD operations were automatically generated as generic SPs lacking business logic. Four stored procedures where generated for each table, one for each CRUD operation. The higher-level custom stored procedures encapsulate custom business logic; yet utilize the generic CRUD SPs for most data manipulation operations. All data manipulation operations were performed using transactions, with appropriate isolation levels, to ensure data integrity. Metrics concerning the stored procedures are covered in following sections.

Also within the database, common language runtime integration was used to implement select encryption algorithms. A common code base was used to compile two libraries. One library targeted the Silverlight runtime and was deployed to the client. The second library target the .NET runtime hosted in SQL Server. Libraries generated from a common code base helped to ensure results calculated in either environment were identical.

Middle Tier

The middle tier was implemented as a Windows Communication Foundation (WCF) Web service hosted within Internet Information Services (IIS). Language-Integrated Query (LINQ) was used almost extensively in this code to standardize access to the database. The database server was logically separate, although physically located on the same machine.

The main operations that constituted the business logic were:

1. User authentication
2. Retrieve communication partners
3. Retrieve recipient public key
4. Send message
5. Retrieve message
6. Retrieve private key

Each one of these operations was implemented as a Web service method. User Authentication occurs for both recipients and senders. Operations 2 through 4 occur on the sender side of a communication event. The last two occur on the recipient side of a communication event. These were implemented much as planned, and can be seen in the system sequence diagrams and use cases.

LINQ is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages (18) through the addition of a general-purpose query facility in the programming language. Practically, the means we avoid embedding Structured Query Language (SQL) strings in source code and submitting them through a database connection at run-time. The database operations now become part of the code and the LINQ provider handles the execution logic. This allows for rich metadata, compile-time syntax checking, static typing and IntelliSense at development time.

In the Cipher-N Web service most data access was accomplished with LINQ. For security reasons, Silverlight does not provide for direct database connectivity. Therefore, any database

interaction is first routed through a call to a Web service method. The Web service methods are then responsible for calls to the database in which cases LINQ queries were utilized.

Presentation Tier

The presentation tier was comprised of the following technologies all working within the Web browser:

- ASP.NET
- JavaScript
- AJAX
- Silverlight

Of the technologies listed, the bulk of the site was written using traditional ASP.NET, AJAX and JavaScript. Virtually all visual elements of the site were HTML-based using cascading style sheets (CSS). Utilization of CSS allows for easy site maintenance and allows the aesthetic aspects of the site to be updated by a Web designer, rather than a software developer.

Authentication to Web site was accomplished using standard HTML text fields in conjunction with an open source JavaScript encryption library (19). Upon submission, the user's password is hashed in the browser using the library's SHA256 implementation. The resulting hash is transmitted to the Web service and compared to the hash on file. This client-side encryption pattern ensures the plain-text password is never transmitted across the wire.

Once authentication a user can choose to securely transmit a file. To send a file a user must first select it from their local file system. For security reasons, Web browsers traditionally run in a "sandboxed" environment with limited access to the host file system. Read access to the clients local file system was accomplished using the standard OpenFileDialog Silverlight control, which ensures appropriate protections for the client's operating system.

Silverlight was mainly employed for its ability to utilize CLR-based encryption libraries and Web service interaction. Of the encryption technologies required by the project, a few were available as part of the base class library in Silverlight 2.0. Specifically, we utilized the Advanced Encryption Standard symmetric algorithm and the SHA256 hashing algorithm implemented in the System.Security.Cryptography namespace (available in Silverlight 2 Beta 1).

A custom encryption library was also created which utilized code from the Mono project (20). The Mono project is a cross platform, open source implementation of the .NET Framework. The Mono

project contains an encryption implementation, RSAManaged, not currently available in the Silverlight BCL. However, as-is the Mono implementations would not work out of the box in the Silverlight runtime. This was due to their use of unsafe code blocks added to improve performance. Sections of the code that utilized unsafe code blocks had to be refactored using Silverlight safe methods. This resulting source code was compiled into two libraries: one for the Silverlight CLR and the other for the SQL Server CLR. This ensures encryption and decryption operates are performed identically on both the client and the server.

To securely retrieve a file, the file is first download to the browser and decrypted in the Silverlight CLR. Ideally at this point, the user would select a location on the local file system where the decrypted file would be saved. However, as of version 2.0, the Silverlight runtime does not provide the controls necessary to gain write access to the client file system. Although not an ideal solution, a temporary workaround implemented. The workaround consists of posting of the decrypted file back to the Web server over HTTPS. A standard Web browser save file dialog is then present to the user allowing them to select a location on the local file system to which to save the file. Again, this solution is not ideal since it involves an additional roundtrip to the server over a standard 128-bit encrypted channel. When the control to securely save files to local disk is added to the Silverlight runtime it will replace the current save functionality.

6.3 Processes

This section reviews the software development processes used in development of this system. First we discuss the goals and technological context for the project. Second we discuss the agile methods that we employed during the development lifecycle. Third we discuss the general sequence of development. Finally we discuss the testing methods used.

Proof of concept

The fundamental goal of the CapStone project was to develop a proof of concept utilizing Silverlight 2 as centerpiece of the development effort. The analysis, documentation, and software framework developed definitively show that the concept is feasible and it forms the basis of a future commercial offering

Spiking

As suggested in Section 3.2 - System Development Lifecycle additional iterations of the UP were completed during the development process. A large portion of the time spent during these iterations involved additional spike tests to assess rapidly emerging technologies including WCF

and LINQ. Additional spiking was also required for the two high risks areas identified in the inception phase - usage of Silverlight and availability of cryptographic algorithms in the Silverlight base class libraries. The following are brief descriptions of the additional test presented by their tier of focus.

Data Tier

To facilitate usage of a common code base by the client and database, managed code services within the database were also investigated. As of version 2005, Microsoft SQL Server integrates the common language runtime component of the .NET Framework. To test this feature of the database two libraries containing sample calculations were compiled from the same code base – one for the Silverlight CLR and one for the SQL Server CLR. The results from both environments were then compared for consistency. Tests conducted performed as expected. However, due to the use of unsafe code in the test calculations additional configuration was required within SQL Server to allow execution of the managed code.

Middle Tier

An emerging technology investigated for use in the middle tier was LINQ to SQL. LINQ to SQL is used to query relational data stores without leaving the syntax or compile-time environment of the local programming language. Several small tests involving each of the CRUD operations were conducted against the Cipher-N database. The tests worked as anticipated with no issues.

Several tests were also conducted to assess Silverlight's ability to consume ASMX Web services and WCF SOAP-based Web services. A Silverlight 2 client can access the data from both Web service technologies by creating a proxy to the service, which generates a class description for the exposed classes. Proxies for both technologies running in standalone and IIS hosted configurations were generated. Tests conducted for each configuration eventually worked; however, both the Alpha and Beta 1 release of the Silverlight SDK surfaced considerable problems with Web service testing. The beta 2 and release versions addressed the outstanding bugs and showed no issues during testing.

Presentation Tier

To facilitate a better user experience, a test for inclusion of the ASP.NET AJAX Control Toolkit (21) was prepared. A minimal web page containing two tabs, each hosting an ASP.NET control

and a Silverlight control, was created. Only a minor issue related to tab index persistence was observed during testing.

Two JavaScript cryptographic libraries, ClipperZ (18) and jsSHA2 (22), were also investigated during the process. A minimal web page containing only text fields and a submission button were created to test each library. Both libraries provided the necessary SHA256 implementations and both provided answers identical to a known good reference. Ultimately, the decision to use the ClipperZ library was based the high likelihood of continued support; given it is a SourceForge hosted project.

Another area that required extensive testing and investigation related to the core cryptographic classes necessary to the project. Testing on symmetric, asymmetric and hashing classes from the Mono project was conducted. Tests on the Mono project source were targeted at determining where it was compatible with the Silverlight CLR. Due to the limited number of libraries available in the Silverlight BCL and limitations on unsafe code, none of the source was compatible as-is. In all cases code modifications were necessary largely due to unsafe code optimizations. Once introduced to the Silverlight BCL in Silverlight 2 Beta 1, issues related to the symmetric and hashing classes became irrelevant.

A final area of testing that involved a considerable amount of time centered on changes in the Silverlight releases. During the development cycle a total of three transitions were necessary: alpha to beta 1, beta 1 to beta 2, and beta 2 to release. Each migration invoked a large amount of testing and each involved making changes to address breaking changes. Of particular, not was the transition from alpha to beta 1, which transitioned support for synchronous Web service calls to exclusively asynchronous calls.

Project Methodology

Throughout the development process some key tenants of the Unified Process and agile best practices were followed, including management of the application lifecycle, risk focus, use case driven functionality, and constant refactoring.

For the critical task of application lifecycle management an instance of Team Foundation Server was deployed. From the project beginning, task assignments for each developer, Charles Laymon, and myself were tracked in the system. This allowed efficient delegation of tasks while maintaining a record of the activity.

Lifecycle management also included heavy utilization of source control, especially during times of transition between Silverlight releases. By agreement, code was checked in often, but changes could not break the build. In other words, only working code should was checked in. At times when the code would not build, but the added safety of source control was desired, a TFS facility called shelving was used. The shelving option essentially backups the code, without committing it to the main project. Current checkouts, the default configuration, were also utilized. On check-in, the option to automatically or manually resolve conflicts is provided. However, given the scope of work defined for each developer, this was rarely the necessary. The bug tracking functionality also served to track and communicate issues, albeit limited due to the defined scope of the project,

Two main areas of risk were also careful watched during development: technology risk and functionality risk. As previously described, technology risk was largely minimized through the use of extensive spike testing and staying true to the use cases mitigated functional risk. In other words, when questions arose during the development cycle, the specification was referenced frequently to help control scope creep.

Constant code refactoring was another main theme in the project. Essentially, this means the code was made to work, and then distilled down to the essential amount of code necessary to solve the task at hand. Each refactoring of the code resulted in a much leaner, easier to read, more maintainable version of what existed prior. To aid with contact refactoring a few key tools were employed: ReSharper and GhostDoc. ReSharper assisted with code cleanup, performing method and variable renaming and general code styling. GhostDoc was used to automatically generate much of the source code documentation.

Sequence of Development

As previously discussed, the practice of iterative and incremental development was followed for much of the coding cycle; thus, at any given point in time, source code implementing each of the tiers was being modified. However, the general sequence of development initially focused on the business tier, followed by the middle tier, and finally the presentation tier.

Initial development activities for the project began with the data tier focusing on development of an abstract conceptual representation of the Cipher-N data in the form of an entity relationship model. During modeling activities, use cases developed in the inception phase were used as the main point of reference for determining the entities and relationships in the system. Other sources, such as existing previous Cipher-N projects, were used to a lesser extent.

Upon completion of the model, stored procedures implementing the basic CRUD operations were automatically generated using a custom T-SQL script. Four stored procedures where generated for each table, one for each CRUD operation. Sample data was then injected into the database using higher-level custom stored procedures, which were constructed using the basic CRUD procedures as building block.

Middle tier development followed the first implementation of the data tier. Early iterations of the middle tier's business logic consisted of a .NET interface class exposed via an ASMX web service. The interface class exposed methods necessary to implement activities defined by the system use cases. Concrete implementations of the interface accomplished data access activities through the use of .NET TableAdapters tied to the database schema.

Following implementation of critical infrastructure, activity shifted to the presentation tier. Early on the presentation tier consisted of a basic ASP.NET webpage hosting a Silverlight control. The control comprised one visual component (a button) for initiating actions and an ADO.NET Data Service proxy for communicating with the web service.

With the basic framework in place, functionality was slowly added to build out the service defined by the use cases. Along the way numerous changes to the underling encryption algorithms, data access technologies, web service deployment technologies, and Silverlight runtime versions were made. However, the overall framework created in the early stages of development remained unchanged.

Testing

During development a number of different testing processes at various levels were utilized, including ad-hoc, black box, and usability testing at the component level and integration testing at the system level.

At the component level, the bulk of the testing consisted of improvised, ad hoc testing designed to measure component functionality in situ. Ad hoc testing included method stubbing to simulate the behavior of yet-to-be-developed code, which was especially useful for simulating the interactions between the Web service and database. For example, sample files and data sets were persisted as base 64 encoded files on the Web server. Methods that ultimately resulted in more complex operations with the database when then stubbed out using the on-disk data as a temporary data source.

Black box testing and usability testing were also utilized at the component level. For many of the cryptographic algorithms results of known outputs from known inputs were compared. The tests were preformed to confirm subtitles in algorithm implementation did not result in conflicting results. For several of the Web interface components usability testing was also performed.

At the system level, numerous rounds of integration tests were perform to evaluate the end-to-end interaction of tiers. The integration tests evaluated the system's compliance with its specified requirements as described by Appendix B Figures 6 and 12 and illustrated in Appendix C Figures 1 and 2. Testing at the system level culminated when the requirements were met.

In general, the testing thus far is best described as “happy path testing” where the test case use known inputs and execute without exception and produce an expected output. Due to the limited scope of the project, testing has not been performed for areas outside of those necessary to accomplish the tasks outline in Section 4 – Scope. For a production system, additional testing for input validation, values outside of the expected range, and system failures should also be conducted.

6.4 System Metrics

Code metrics to date for each tier of the Cipher-N project are as follows:

Data Tier

Assembly	# Types	# Abstract Types	# IL instruction	# lines of code	# lines of comment	% comment
Stepquest.CipherN.Dml v1.0.0.0	18	2	9721	1483	1141	43%

Cipher-N Database Schema

- Total number of tables: 70
- Total number of stored procedures: 286
- Total number of functions: 3

Middle Tier

Assembly	# Types	# Abstract Types	# IL instruction	# lines of code	# lines of comment	% comment
*App_Code v1.0.0.0	22	0	3456	247	416	62%
Stepquest.CipherN.Components v1.0.0.0	48	0	12682	1688	294	14%

Cipher-N Web Service

- Total number of web methods: 4

Presentation Tier

Assembly	# Types	# Abstract Types	# IL instruction	# lines of code	# lines of comment	% comment
Stepquest.CipherN.WebClient v1.0.0.0	20	2	3284	452	266	37%
Stepquest.CipherN.WebClientComponents v1.0.0.0	25	2	12345	1894	1099	36%
*App_Code v1.0.0.0	22	0	3456	247	416	62%

Cipher-N Website

- Total number of ASP.NET pages: 4
- Total number of custom ASP.NET web user controls: 6
- Total number of Silverlight controls: 1

*Includes Cipher-N ASP.NET web application code and Stepquest.CipherN.WebService namespace.

Full Cipher-N Application

A significant amount of time was required to successfully design and implement the Cipher-N proof of concept. During the project's time frame (July 2007 – December 2008), approximately 564 hours of research and development time were required. Also, during this time a total 358 commits were made to the source control system. The following diagram illustrates the completed system's components and dependencies.

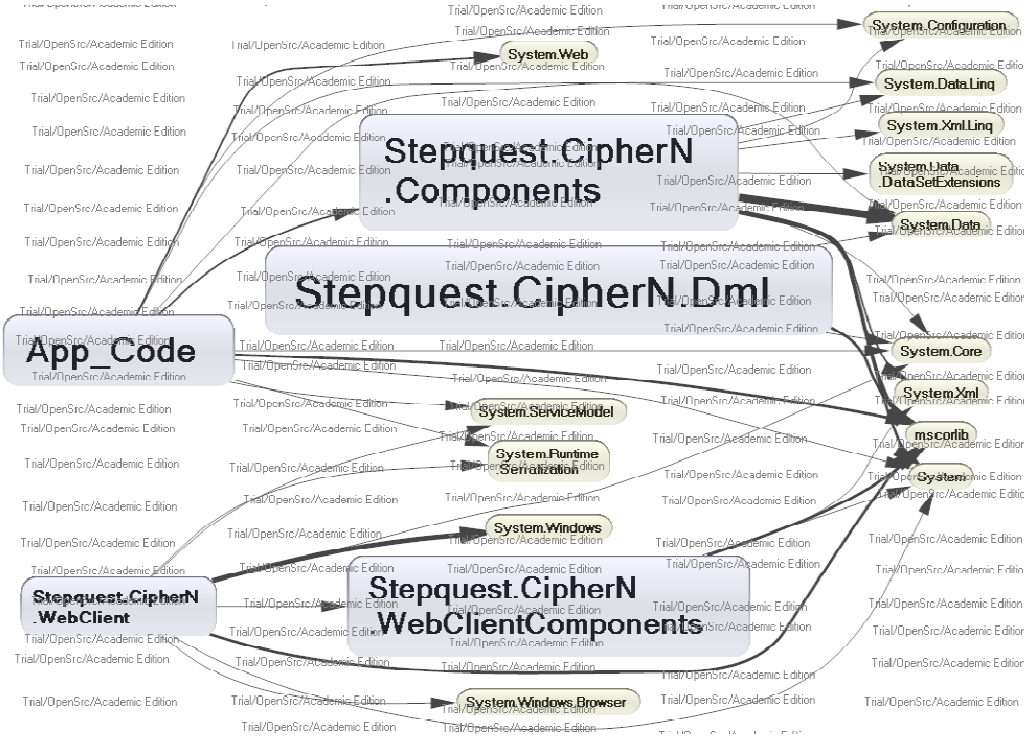

Figure 6.2 - Component Dependencies

6.5 Critical Analysis

Motivation for the Cipher-N project stemmed largely from the desire to apply new methodologies and technologies to a real work project and to provide information back to the development community based on that experience. Included amongst the goals of the project was the desire to investigate the merits of the UP for a small scale, small team project of this type. Additionally, several new technologies were to be investigated in order to answer questions such as: In what situations do the new technologies make sense and what features helped achieve greater productivity? A final goal was to provide a better understanding of the trade-offs for utilizing highly abstracted data models.

Unified Process

As outlined in Section 3, an iterative and incremental development process known as the Unified Process was the chosen development methodology for the project. Having now completed the project, I discuss the advantages and disadvantages as I perceived them.

Advantages

Phases of the UP strive to achieve number goals (see Figure 3.2 - UP Phases and Objectives). I found the end product of many of the defined goal to be extremely useful for this project's development.

Two goals achieved early in the process were particularly helpful. One goal was to define project scope and boundary conditions. Achieving this took several in-depth with the client, since early in the project only a high-level idea of the requirements had been discussed. Each meeting helped to surface issues and refine requirements. Once I began flesh out the details it became obvious that numerous process and technology issues were present. It also became readily apparent that the project was not a cookbook application of known technologies; rather it was a proof-of-concept with many unknowns. Such discoveries, help to achieve a second goal, that of addressing high risk areas early in the process.

A third goal, use case development, also proved to be very useful; the most useful of all the goals, in my opinion. This was especially the case once we began to map out the system interactions for end-to-end encryption. During this time, a number of misunderstandings with regards to cryptographic algorithm classes were discussed. Use case development also forced discussion of a key system security feature, recipient-only decryption, which had a major impact on overall system implementation.

The risk focused nature of the UP also had a positive impact on the project. Early in the development cycle major technology risks were identified. The risks were largely related to stability and completeness of early cycle products and the major role they would play in the final product. By addressing these concerns early, through spike testing and demonstration of a stabilized baseline environment, we were able to move forward with confidence in the project's success. Further, it helped to achieve another goal, realistic schedule estimation. Additionally, having identified several risks early we likely avoided significant rework later in the development cycle.

An unforeseen benefit of the process came later in the development process. During time of coding a found the use cases related to system interaction for encryption and decryption to be excellent points of reference. Having visualized the system through use cases, I was able to reason about the cryptography processes form a macro and micro level with relative ease.

Disadvantages

In my option, the disadvantages of UP come from two areas: upfront time required before development commences and the need to document the obvious.

The time required for both the client and myself to develop a complete set of use case was significant. Contrary to most real world projects, time pressure for the Cipher-N project was minimal. As such, I was able to spend the time and effort necessary to develop a through set of use cases. Applying this in the work place may be difficult due to the “perceived” lack of progress (e.g. no lines of code) while early UP phases are completed.

The need to document obvious is also a weakness of the process. As I mentioned previously, use case mapping was extremely useful for many interactions, especially complex ones. However, other interactions such as a user logging onto the system or send a user notification likely do not require the same scrutiny. Execution and implementation of these common interactions should be obvious to a professional developer.

New Technologies

As alluded to in previous sections, a large portion of development time was spent investigating several new technologies ultimately used by the project. In the project proposal the technologies identified for investigation were Silverlight 2.0, Visual Studio 2008, and Visual Studio Team System 2008. During the development cycle a few additional technologies were added to the list. These include SQL Server 2008 and LINQ.

Silverlight and Visual Studio 2008

Overall, Silverlight development was relatively easy to learn. However, as would be expect from an alpha cycle release, initial development utilizing the Silverlight tools integrated with Visual Studio was a bit challenging. IDE crashes occurred frequently, libraries references did not

consistently resolve, and Web service proxy generation worked infrequently at best. Further, support for cryptography algorithms was not existent and web service calls were buggy.

With progressive iterations of Silverlight (currently Silverlight 2.0 RTW) integration and performance issues lessened. To date all previously described issues have been resolved. Silverlight's support for several of the .NET languages and close ties to WPF allowed me to leverage my existing skills and quickly come up to speed. Built-in project templates targeting Silverlight applications and libraries made it quick and easy deploy spike tests. Furthermore, the end-to-end debugging capabilities proved to be invaluable.

The features in Silverlight were sufficient for addressing the technical requirements of the project. For example, Web service communication by the web client was accomplished solely through the use of the automatically generated, strongly typed Web service proxies. Hashing and symmetric cryptography operations were performed using classes provided the BCLs. Client-side LINQ was also used to accomplish many of the data manipulation operations. As a whole, when executing a Silverlight application, this rich set of libraries provided in the runtime leads to a leaner download requirement on the client.

Silverlight 2.0 is not without limitations. Most notably is its lack of a file dialog for saving files to the local host. For the Silverlight 2.0 release this functionality was intentionally excluded for "security reasons". Microsoft has unofficially acknowledged that such a control will be available in a future version. In this meantime, users are left to implementing workarounds such as the one used in the Cipher-N project.

When dealing with a mixed solution projects containing both ASP.NET AJAX and Silverlight, executing managed code from JavaScript or invoke JavaScript objects from within managed code is a must. In Silverlight vernacular, this functionality is referred to as the HTML Bridge. Among other things, the bridging functionality provides a mechanism to obtain managed references to DOM elements. Unfortunately, the managed objects do not directly carry information regarding the properties and methods exposed by the DOM element. API lookups were required to obtain this information. Hence, I found debugging these interactions to be extremely challenging.

Other areas lacking in the Silverlight environment included its ability to support asymmetric cryptography, its ability to support direct database connections, and the extremely limited modal dialog (added in Silverlight 2.0 RTW).

Team System Integration with Visual Studio 2008

Visual Studio Team System 2008 was the application lifecycle management tool utilized for the project. Although it provides many other services, it was mainly utilized for its source control, work item tracking and bug tracking capabilities. In this capacity it performed beyond expectations. The tight integration with Visual Studio allowed me to work almost exclusively in the IDE. Of particular importance was the ability to compare previous versions of a file to the current version. Further it was often necessary to request assistance from the other developer on a particular issue. Having a common source repository made it extremely easy for this to occur. The added capability to track and communicate tasks with the other developer was also particularly useful during the projects early iterations.

LINQ

Another promising technology investigated during development is known as LINQ. LINQ is .NET framework component used to query relational data stores without leaving the syntax or compile-time environment of the local programming language. It also allows developers to optionally transform/shape data query results into whatever format they want, and then easily manipulate the results.

Development with the LINQ technology was a natural fit for accessing data from the middle tier. In the case of Cipher-N, middle tier functionality receives requests via the Web service interface which interacts with the database to service the request. A potential solution could have utilized TableAdapters or a Data Provider for SQL server. In either case, embedded SQL or stored procedures would need to be generated. Utilizing the LINQ, the database connection and necessary SQL to manipulate the data are generated on the fly. Overall, this amount of custom code required to accomplish the task was kept to a minimum with no loss in functionality.

LINQ does however have some downsides. Among them is the moderately steep learning curve required to understand the syntax. Understanding exactly what is happening with a chain of Lamba expressions can be somewhat challenging and unintuitive. The ability to directly manipulate and test SQL statements directly in SQL Server Management Studio is also detracting. Although, an available Visual Studio LINQ debugger visualizer helps migrate this issue. A final consideration is performance. Since, LINQ builds it quires on the fly, it does not offer the added benefit of a cached execution plan, as would be the case with a stored procedure.

SQL Server 2008

Cipher-N project development began on SQL Server 2005; however, the database was migrated following the release of SQL Server 2008 RC. The transition was made for several reasons including: enabling access to the .NET 3.5 framework (only 2.0 was available in SQL Server 2005), access to a new feature called FILESTREAMS, which is used for managing semi-structured data (e.g. documents) and access to the new SQL Server Management Studio that provides IntelliSense for SQL queries.

In general, development on Microsoft SQL Server 2008 was very similar to 2005. Performance of the new SQL Server Management Studio was, however, vastly superior to 2005 in terms of response time and productivity. The increased response times dramatically reduced the time necessary to display schema data and properties. Such times may be inconsequential when connecting to a database on the LAN. Connecting to a remote database, as was the case here, the improvement was extremely beneficial. Productivity gains experienced were due in large part to the addition of IntelliSense.

Deployment and utilization of .NET assemblies to the database was easy and useful. For instance, a requirement of the project dictated password hashing in both the Silverlight client and database. Implementing the hashing algorithm in both places could result in errors and future maintenance issues. Utilizing a Visual Studio database project I was able to code, build and deploy an assembly directly to the database. This tight level of integration allowed me to compile two libraries from the same code base. One was deployed to Silverlight and the other to the .NET in the database.

Abstracted Data Model

Described in Section 3.3, the database design employed by the project employed an entity model abstracted to a higher degree than typical applications of this size. The reason for this is twofold. First, the schemas for some of the entities (a logical grouping of related tables) were imported from other StepQuest projects. Second, there existed a desire to build-in a degree of flexibility for future application expansion.

The challenge in working with the model stemmed from the fact the numerous DML operations were required in some cases to manipulate small amounts of data. Additionally, the large number

of tables and the type/super type nature of the design, inherit in the model's abstraction, contributed to the difficulty in conceptualizing the data model.

As development progressed the model became more effective to use. Custom stored procedures which severed as "the glue" holding together calls to generic SPs were created; thus, making data manipulation far easier. Grouping of tables as entities made it easy to reason about specific areas of the model and the auto-generated CRUD SPs were useful for many data operations. The flexibility inherent in the design also made addition of attributes and/or slight model changes trivial.

7 Summary and Future Work

In summary, it was learned that the concept of a client-side, browser-based and client-platform agnostic encryption service, which utilizes Microsoft's Silverlight as the encryption engine, is a viable concept for commercial implementation. There are, however, limitations in the Silverlight technology (e.g., local file saving) that will need to be addressed prior to such an endeavor. It was also learned that the development process and technologies investigated for the project were adequate to meet the needs of the project and in my option, are useful and productive. Finally the projects lead to a much better understanding the requirements and use case support required to implement a commercial offering. In the end, the knowledge gained throughout the project was provided the customer, StepQuest, as a complete set of design and analysis documentation.

During the development of the Cipher-N project concepts and features outside of the current system scope were also discussed. The concepts and features recommended for future investigation include:

- Exploration of assembly obfuscation technologies. Obfuscation of the deployed .NET assemblies would lead to greater application security by reducing the risk of decompilation and reverse engineering.
- Hardening of the Cipher-N web service by ensuring its surface of vulnerability no longer than that required to support the service. The Web service source code and deployment configuration should also be evaluated against the WCF Security Guidance published by Microsoft.
- Implementation of Cipher-N as client-based operating system service which utilizes Microsoft's Live Mesh synchronization technology to securely replication files amongst groups.
- Removal of the client-side symmetric document encryption in favor of asymmetric only encryption. Such a change would require implementation of an asymmetric cipher capable of executing in the Silverlight runtime in a time acceptable for production use.

- Hardening of the Silverlight application through implementation of the SecureString class currently only available in the full .NET framework. The main functionality of the class is to encrypt strings (e.g. password) in when being used, and deleted from computer memory when they are no longer needed. Currently, strings are stored in clear text on the CLR heap.

8 Bibliography

1. **Info-Tech Research Group.** Info-Tech Research Group: IT Security Spending by U.S. Companies Will Hit US$61 Billion for 2006, Says Info-Tech Research Group. *infotech.com.* [Online] [Cited: January 1, 2008.] http://www.infotech.com.

2. **PGP Corporation and Vontu, Inc.** *2006 Annual Study: Cost of a Data Breach.* s.l. : The Ponemon Institute, 2006.

3. **Hush Communications.** "About". [Online] Hush Communications. [Cited: November 18, 2007.] http://www.hushmail.com/about.

4. **Arrington, Michael.** "a-comparison-of-live-hotmail-gmail-and-yahoo-mail". [Online] Feburary 8, 2007. [Cited: November 18, 2007.] http://www.techcrunch.com.

5. **Postel, J. and J.Reynolds.** "Rfc959". [Online] IETF. [Cited: November 18, 2007.] http://www.ietf.org/rfc/rfc959.txt.

6. **Die.Net.** "Email". [Online] Die.Net. [Cited: November 18, 2007.] http://dict.die.net/email/.

7. **Callas, J.** "Rfc4880". [Online] IETF. [Cited: December 11, 2007.] http://www.ietf.org/rfc/rfc4880.txt.

8. **Microsoft Corporation.** "What is Microsoft Office SharePoint Server". [Online] [Cited: December 11, 2007.] http://www.microsoft.com/sharepoint/prodinfo/what.mspx.

9. **ISO/IEC.** Information technology — Common Language Infrastructure (CLI) Partition I to VI. [Online] http://standards.iso.org/ittf/PubliclyAvailableStandards/c042927_ISO_IEC_23271_2006(E).zip.

10. **Microsoft Corporation.** "Silverlight Overview". [Online] Microsoft Corporation. [Cited: November 25, 2007.] http://msdn2.microsoft.com/en-us/library/bb404708.aspx.

11. **Adobe Systems Incorporated.** "Features". [Online] [Cited: December 11, 2007.] http://www.adobe.com/products/flashplayer/productinfo/features/.

12. **Sun Microsystems.** "Index". [Online] Sun Microsystems. [Cited: November 25, 2007.] http://java.sun.com/javase/technologies/index.jsp.

13. **The Eclipse Foundation.** "Overview". [Online] The Eclipse Foundation. [Cited: November 26, 2007.] http://www.eclipse.org/jdt/overview.php.

14. **Satzinger, John, Jackson, Robert and Burd, Stephen.** *Object-Oriented Analysis and Design with the Unified Process.* s.l. : Course Technology, 2004.
15. **Newkirk, James and Vorontsov, Alexei.** *Test-Driven Development in Microsoft .NET.* s.l. : Microsoft Press, 2004.
16. **Beck, Kent.** *Test-Driven Development by Example.* s.l. : Addison Wesley, 2003.
17. **Microsoft Corporation.** "N-Tier Data Application Overview". [Online] [Cited: December 11, 2007.] http://msdn2.microsoft.com/en-us/library/bb384398(VS.90).aspx.
18. **Microsoft Corporation.** "LINQ: .NET Language-Integrated Query". [Online] [Cited: November 8, 2008.] http://msdn.microsoft.com/en-us/library/bb308959.aspx.
19. **SourceForge.** “Javascript Crypto Library". [Online] [Cited: November 8, 2008.] http://sourceforge.net/projects/clipperzlib.
20. **Novell.** “Mono". [Online] [Cited: November 8, 2008.] http://www.mono-project.com.
21. **CodePlex.** “ASP.NET AJAX Control Toolkit". [Online] [Cited: November 8, 2008.] http://www.codeplex.com/Wiki/View.aspx?ProjectName=AjaxControlToolkit.
22. “OpenSource JavaScript implementation of the Secure Hash Algorithms, SHA-256-384-512". [Online] [Cited: November 15, 2008.] http://anmar.eu.org/projects/jssha2/.

Appendix A

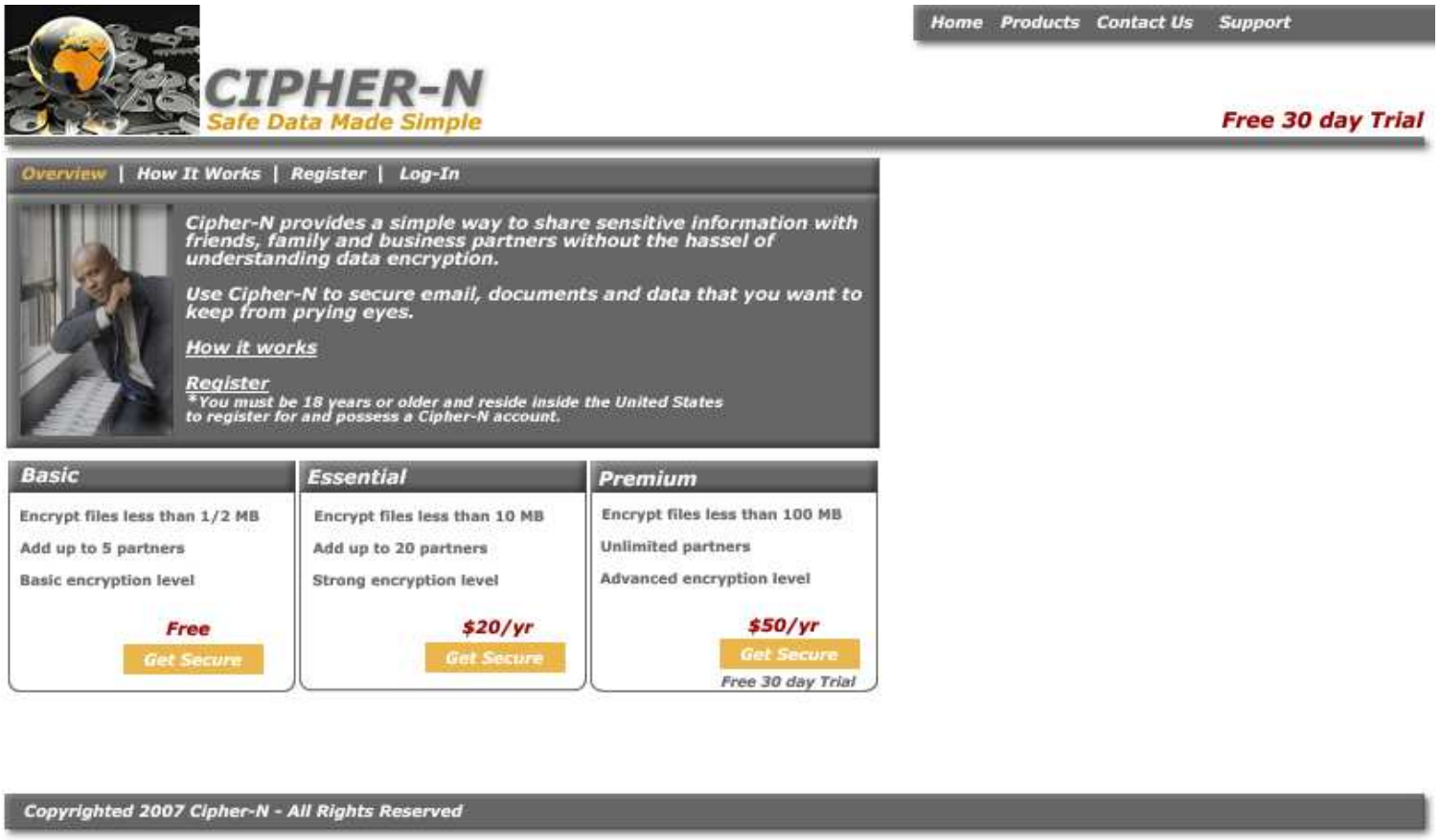

Figure 3 - Home

Figure 4 - How It Works

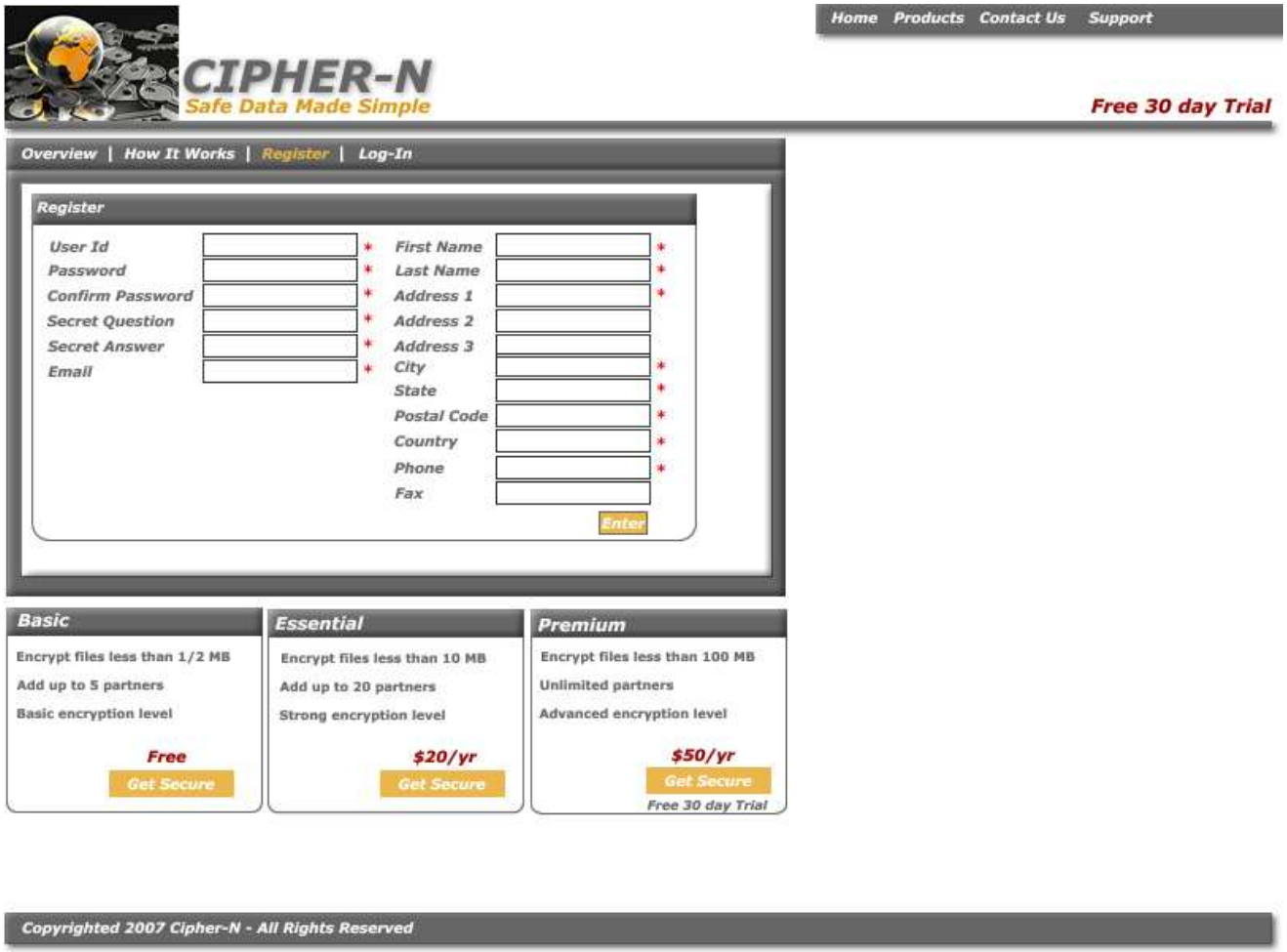

Figure 5 - Register

Figure 6 - Login

Figure 7 - Lost Account Information

Figure 8 - Administrate Profile

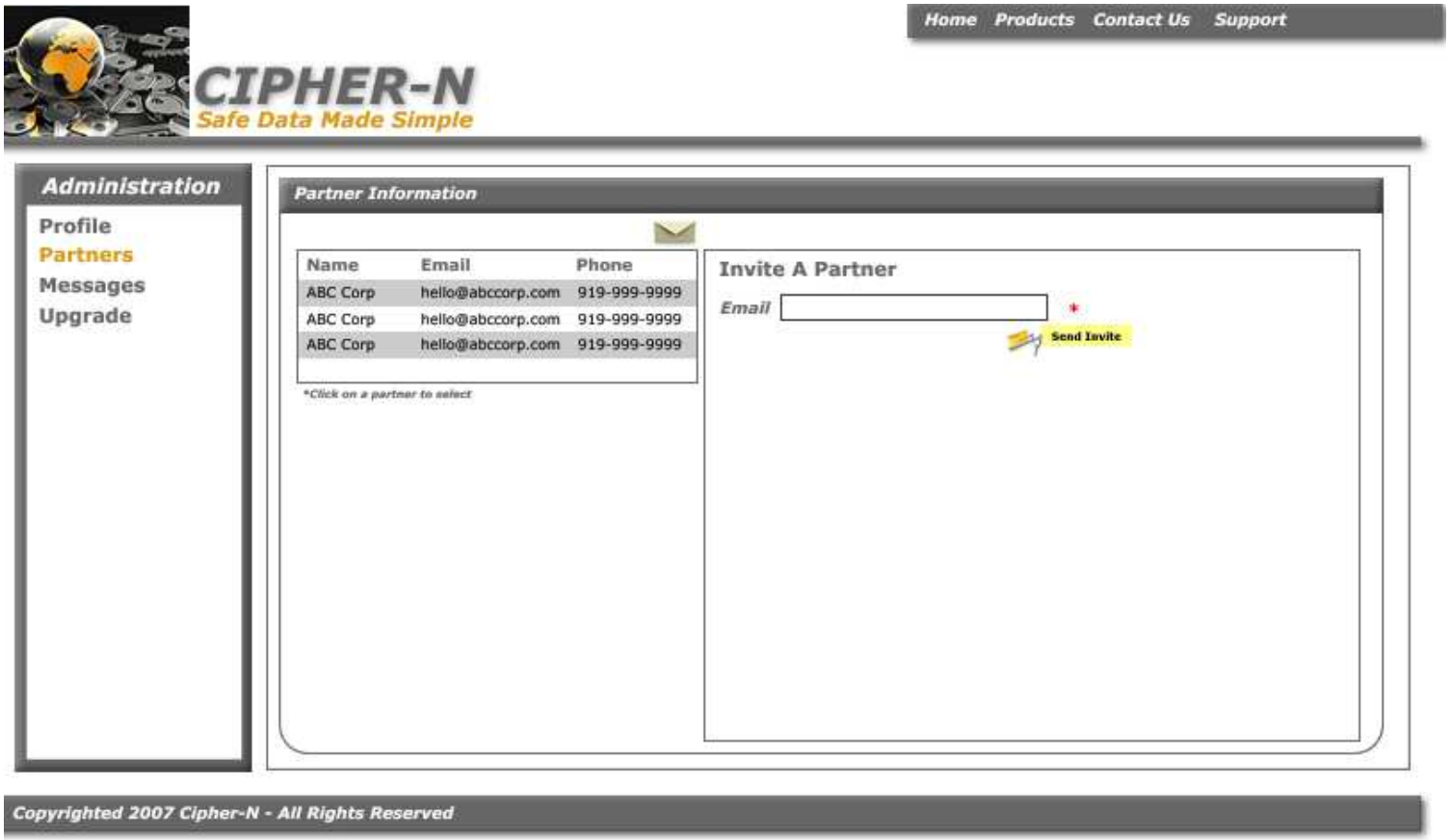

Figure 9 - Administrate Partner

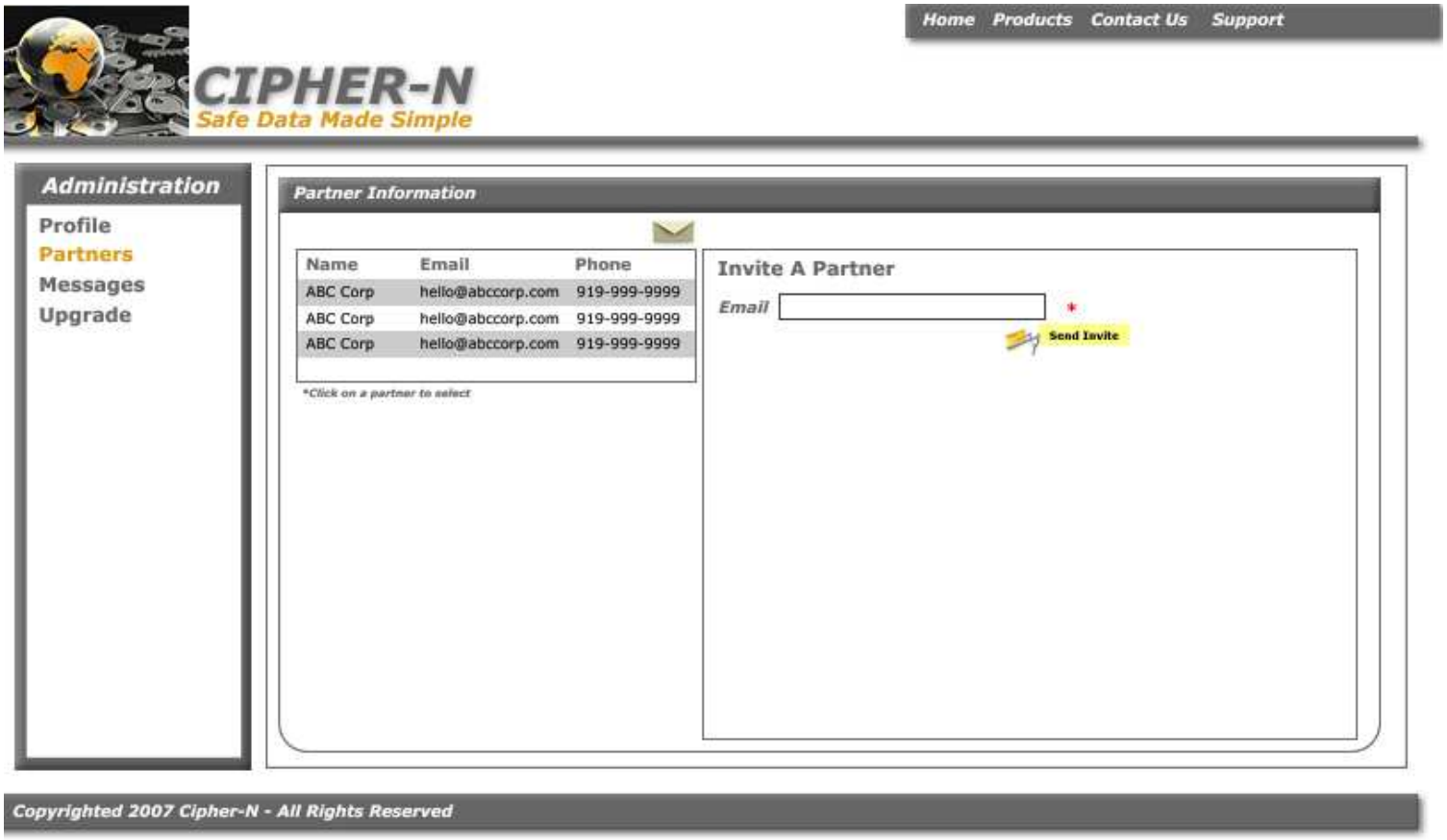

Figure 10 - Add Partner

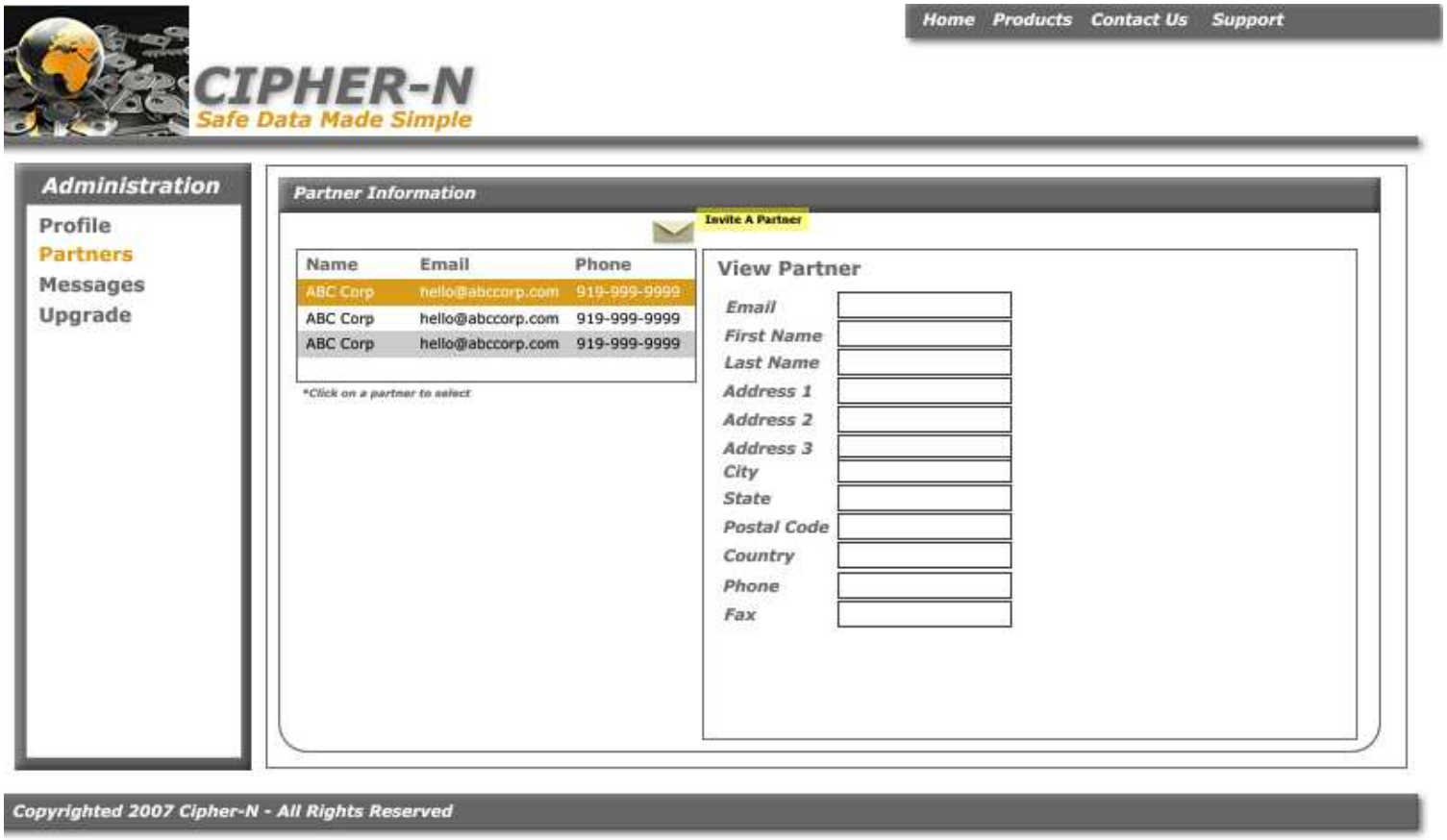

Figure 11 - Select Partner

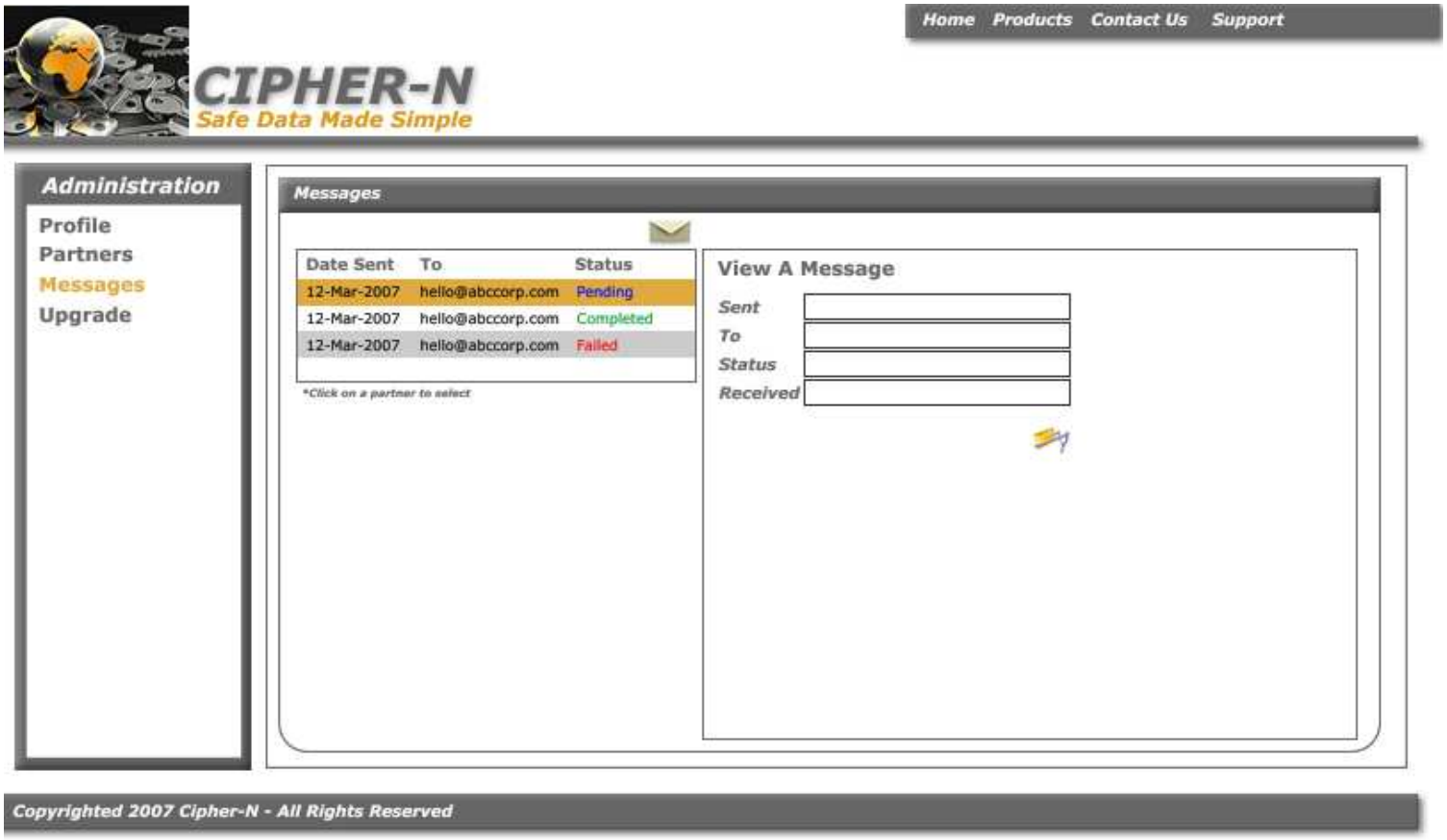

Figure 12 - Message View

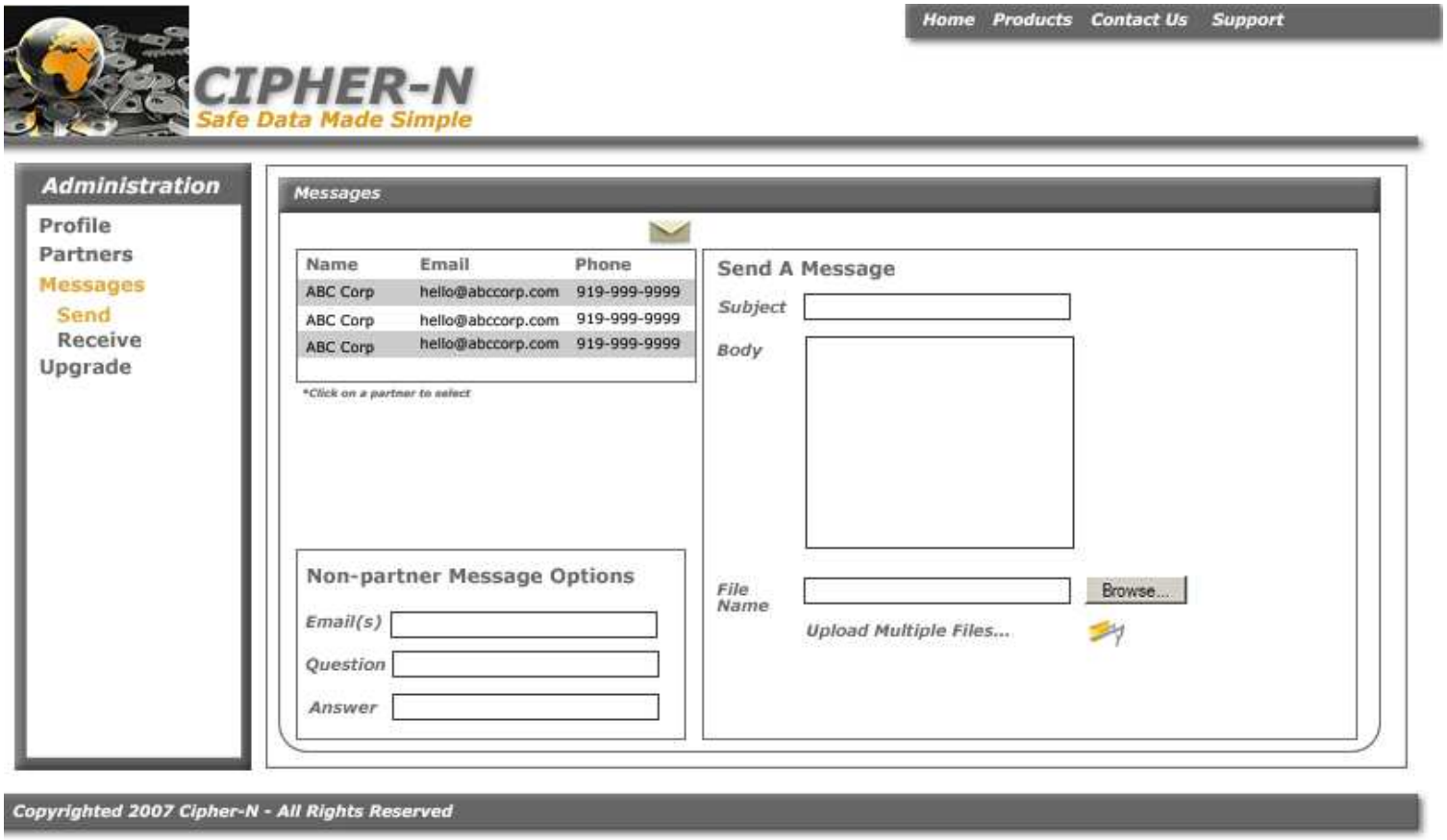

Figure 13 - Send Message

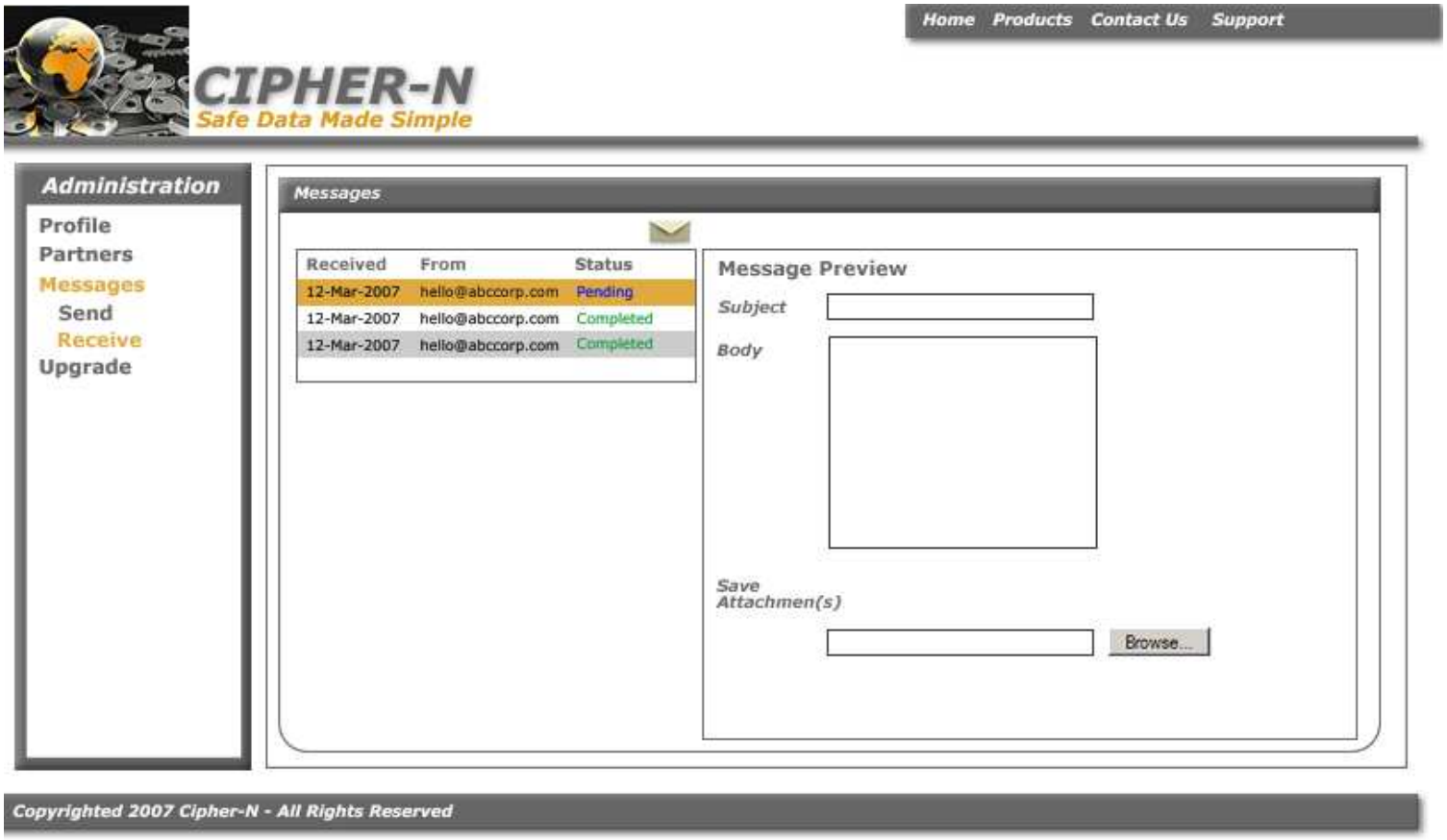

Figure 14 - Retrieve Message

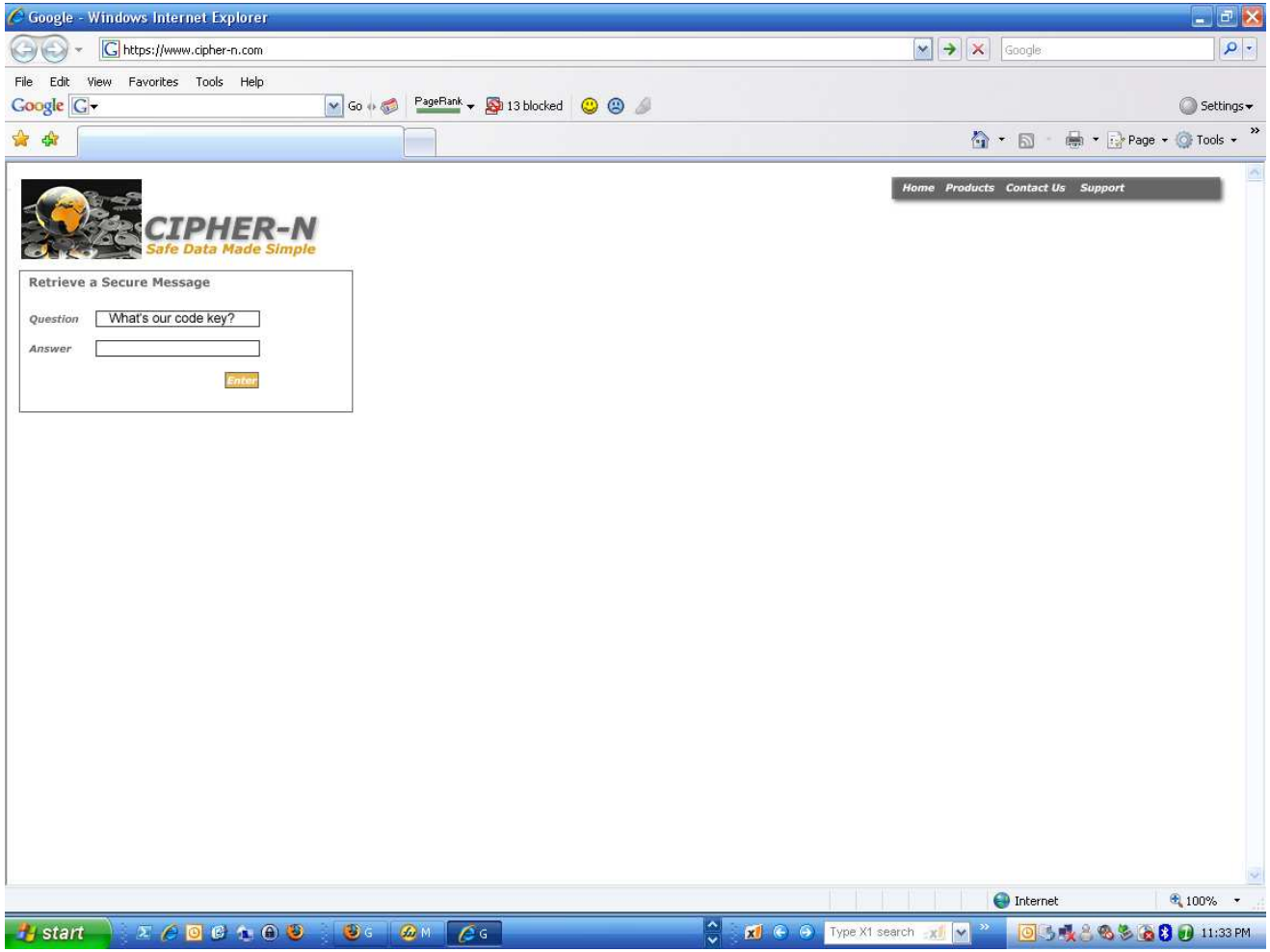

Figure 15 - Retrieve Message Non-Partner

Figure 16 - Change Account Status

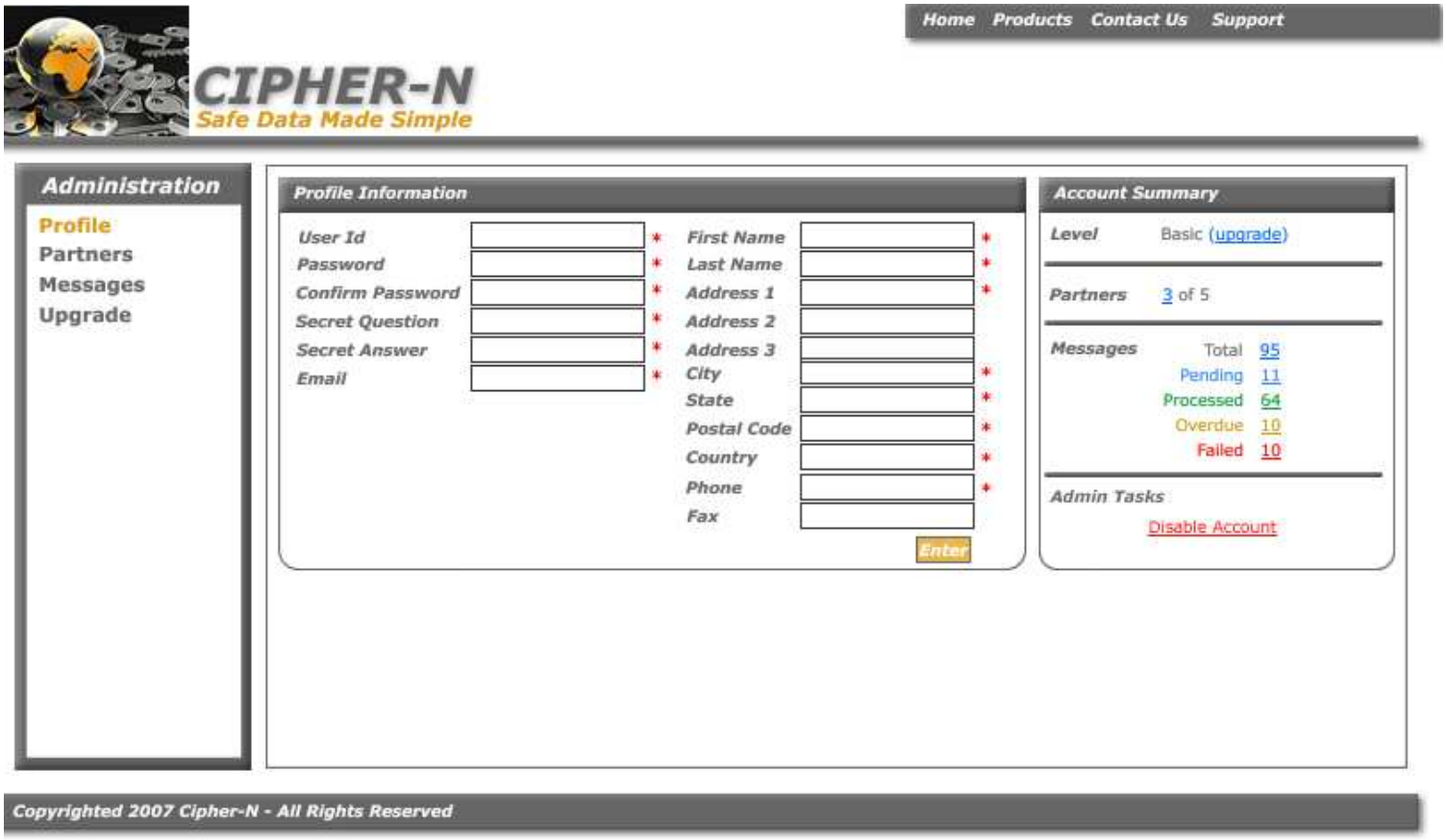

Figure 17 - Modify Account

Appendix B

System Event Table

Id	Use Case*	Event	Trigger	Source	Response	Destination
1	Register new user	Customer creates a new account	Account activation	Customer	Account confirmation email	Customer
2	Authenticate user	Customer authenticates to web site	Account authentication	Customer	Welcome screen	Customer
3	Add new partner	Customer adds a new partner	New partner	Customer	Partner notification email	Partner
4	Query customer partners	Customer views partners	Partner inquiry	Customer	Partner details	Customer
5	Encrypt new message	Customer encrypts a new message	New message	Customer	Encrypted message	Customer
6	View message status	Customer (or administrator) views message status	Status inquiry	Customer or administrator	Message status details	Customer or administrator
7	Upgrade account service level	Customer upgrades service level	Service upgrade	Customer	Upgrade confirmation email	Customer
8	Update customer profile	Customer updates profile	Profile update	Customer	Profile modification transaction	System
9	Lost account information	Customer lost account information	Lost information	Customer	Account information email	Customer
10	Cancel Account	Customer cancels account	Account cancellation	Customer	Profile disable transaction	System
11	Decrypt message (partner)	Partner decrypts received message	Message decryption	Partner	Decrypted message	Partner
12	Decrypt message (non-partner)	Non-partner decrypts received message	Selecting notification hyperlink	Non-partner recipient	Decrypted message	Non-partner recipient
13	Produce summary report	Time to produce activity reports	"End of month"	Temporal	Summary report	Customer or administrator
14	Charge customer account	Time to bill customer	"End of month"	Temporal	Billing transaction	Customer and bank
15	Modify customer account	Administrator modifies a customer account	Account modification	Administrator	Account modification transaction	System
16	Delete account	Customer deletes account	Account modification	Customer	Profile deletion transaction	System

*A use case describes an EBP

Elementary Business Process (EBP): a task performed by one person in one place, in reponse to a business event, which adds measureable business value and leaves the system and its data in a consistent state.

Figure 18 - System Event Table

Use Case Name:	Register new user	
Scenario:	Customer creates for new account	
Triggering Event:	Customer selects "register" button on website	
Webpage Mock-up:	300CipherN_Register.png	
Brief Description:	When a customer selects the "register" menu item on the website, a dialog is presented to obtain customer information. Once submitted, the new account details and a confirmation link are emailed to the customer.	
Actors:	Customer	
Related Use Cases:	Includes: *Lost account information*	
Stakeholders:	Sales department: to collect customer statistics	
Preconditions:		
Postconditions:	New account must be created. Account information must be emailed. The confirmation link sent to the customer must be used to activate the account.	
Flow of Events:	**Actor**	**System**
	1. Customer browses to CipherN web site and selects the "register" menu item. 2. Customer completes required fields. 3. Customer submits information by selecting "enter" button. 4. Customer clicks on hyperlink in email.	1.1 Display data entry dialog. 3.1 Validate data and presence of required fields. 3.2 Create new account. 3.3 Generate email with account details and confirmation link and send to customer. 4.1 Activate account.
Exception Conditions:	3.1 If data is not valid or all required fields are not present, then present error dialog to customer. 3.1 If user id is taken, then invoke *Lost account information*.	

Figure 19 - Register New User

Use Case Name:	Authenticate user	
Scenario:	Customer logs on to website	
Triggering Event:	Customer enters credentials on website	
Webpage Mock-up:	400CipherN_Login.png	
Brief Description:	When a customer enters their logon credentials and CAPTCHA response they are vaildated against the database. If authenticationed, a welcome page is displayed; otherwise, options to re-enter credentials or obtain lost account information are presented.	
Actors:	Customer	
Related Use Cases:	Includes: *Lost account information* Relies on: *Register new user*	
Stakeholders:	Administrators: to monitor site security	
Preconditions:	The customer account must exist and have been activated.	
Postconditions:	N/A	
Flow of Events:	**Actor**	**System**
	1. Customer browses to CipherN web site and selects the "log in" menu. 2. Customer enters credentials and current CAPTCHA word. 3. Customer submits credentials by selecting "enter" button	1.1 Display logon dialog. 3.1 Validate data ard presence of required fields. 3.2 Display welcome page.
Exception Conditions:	3.1 If data is not valid or all required fields are not present, then present error dialog to customer. 3.1 If 3rd failed authentication attempt, then invoke *Lost account information* 3.1 If 10th failed authentication attempt, disable account	

Figure 20 - Authenticate User

Use Case Name:	Add new partner	
Scenario:	Customer adds a new partner	
Triggering Event:	Customer enters a new partner's information on the website	
Webpage Mock-up:	510CipherN_Admin_Partner.png	
Brief Description:	When a customer selects the "partners" menu on the web site, a dialog is presented containing their current partners and fields for new partner data. Once submitted, the new partner is emailed an invitation.	
Actors:	Customer	
Related Use Cases:	Relies on: *Authenticate user*	
Stakeholders:	Sales department: to collect customer statistics	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case.	
Postconditions:	Invitation must be emailed to partner.	
Flow of Events:	Actor	System
	1. Customer logs in to web site and selects the "partners" menu. 2. Customer completes required fields for new partner. 3. Customer submits information by selecting the "Invite Cipher-N Partner" button.	1.1 Display current partners and fields for new partner information. 3.1 Validate data and presence of required fields. 3.3 Create new partner. 3.4 Generate invitation email and send to new partner.
Exception Conditions:	3.1 If data is not valid or all required fields are not present, then present error dialog. 3.1 If partner already exists, then present error dialog.	

Figure 21 - Add New Partner

Use Case Name:	Query customer partners	
Scenario:	Customer views list of available partners and their associated details	
Triggering Event:	Customer selects "partners" menu on the website	
Webpage Mock-up:	530CipherN_Admin_Partner_Select.png	
Brief Description: Actors:	When a customer selects the "partners" menu on the website, a dialog is presented containing their current partners. By selecting a particular partner entry, the customer can view additional details about that partner. Customer	
Related Use Cases:	Relies on: *Authenticate user* Relies on: *Add new partner*	
Stakeholders:	N/A	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case. Customer has previously added a partner(s) to their account by invoking the *Add new partner* use case.	
Postconditions:	N/A	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "partners" menu. 2. Customer selects an existing partner.	1.1 Display partners w/ minimal details. 2.1 Display partner w/ full details.
Exception Conditions:	1.1 If no partners exist, then present a message indicating so.	

Figure 22 - Query Customer Partners

Use Case Name:	Encrypt new message	
Scenario:	Customer encrypts a message to exchange with a partner	
Triggering Event:	Customer uploads new message to the web site for encryption	
Webpage Mock-up:	550CipherN_Admin_Send_Message.png	
UML Artifacts:	CipherNSequenceDiagrams.vsd - EncryptPartnerMessage, EncryptNonPartnerMessage	
Brief Description:	When a customer selects the "messages" menu on the website, a dialog is presented containing their current partners and fields for encrypting a new message.	
Actors:	Customer	
Related Use Cases:	Relies on: *Authenticate user* Relies on: *Add new partner*	
Stakeholders:	Administrators: to monitor site security	
Preconditions:	Customer has successfully authenticated to the web site by invoking the *Authenticate user* use case. Customer has added a partner(s) by invoking the *Add new partner* use case or a question and answer have been provided for non-partner recipients.	
Postconditions:	Encryption key associated with each partner(s) are stored in the database. Email must be sent to recipient(s).	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "messages\send" menu. 2. Customer selects partner(s) for message exchange. 3. Customer enters email, question and answer for non-partner recipients. 4. Customer selects a button allowing them to browse for file(s) on the local system. 5. Customer locates file(s) and initiates upload. 6. Customer selects button to initiate sending message to recipients.	1.1 Display current partners and fields for encrypting a new message. 4.1 Display file dialog. 5.1 Display upload progress. 6.1 Generate email and send to recipients.
Exception Conditions:	2.2, 4.1, 5.1 If package level limits are violated, then present option to upgrade. 4.1 If file dialog is unable to access local drive(s), then present error dialog. 6.1 If customer cancels sending, then discard message.	

Figure 23 - Encrypt New Message

Use Case Name:	View message status	
Scenario:	Customer (or administrator) views status of exchanged messages	
Triggering Event:	Customer (or administrator) selects "messages" menu on the website	
Webpage Mock-up:	540CipherN_Admin_Messages_View.png	
Brief Description:	When a customer (or administrator) selects the "messages" menu on the website, a dialog is presented containing a list most recent messages sent along with their current status. By selecting a particular message, the customer can view additional details.	
Actors:	Customer, Administrator	
Related Use Cases:	Relies on: *Authenticate user* Relies on: *Encrypt new message*	
Stakeholders:	Sales department: to collect customer statistics	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case. Customer has previously exchanged a message(s) by invoking the *Encrypt new message* use case.	
Postconditions:	N/A	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "messages" menu. 2. Customer selects message.	1.1 Display list of most recent messages sent along with their current status. 2.1 Display complete message details.
Exception Conditions:	N/A	

Figure 24 - View Message Status

Use Case Name:	Upgrade account service level	
Scenario:	Customer upgrades account service level	
Triggering Event:	Customer selects the "upgrade" menu under the Account Summary section of the website	
Webpage Mock up:	500CipherN_Admin_Profile.png	
Brief Description:	When a customer selects the "upgrade" menu on the website, a dialog is presented containing the additional service levels available.	
Actors:	Customer	
Related Use Cases:	Relies on: *Authenticate user*	
Stakeholders:	Sales department: to collect customer statistics	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case.	
Postconditions:	Confirmation email must be sent to customer.	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "profile" menu. 2. Customer selects the "upgrade" menu under the Account Summary section. 3. Customer selects new service level.	1.1 Display current profile and account summary information. 2.1 Display list of additional service levels available and return option. 3.1 Display upgrade confirmation. 3.2 Send confirmation email customer.
Exception Conditions:	3.1 If customer select "return" option, then return to display from (1.1).	

Figure 25 - Upgrade Account Service Level

Use Case Name:	Update customer profile	
Scenario:	Customer updates information in account profile	
Triggering Event:	Customer selects the "profile" menu on the website	
Webpage Mock-up:	500CipherN_Admin_Profile.png	
Brief Description:	When a customer selects the "profile" menu on the website, a dialog is presented containing the current profile information. Select profile fields are editable allowing customers to update their profile information.	
Actors:	Customer	
Related Use Cases:	Relies on: *Authenticate user*	
Stakeholders:	N/A	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case.	
Postconditions:	Confirmation email must be sent to customer.	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "profile" menu. 2. Customer updates editable fields. 3. Customer selects "enter" button to save changes.	1.1 Display current profile and account summary information. 3.1 Validate data and presence of required fields. 3.2 Send confirmation email customer.
Exception Conditions:	3.1 If data is not valid or all required fields are not present, then present error dialog to customer.	

Figure 26 - Update Customer Profile

Use Case Name:	Lost account information	
Scenario:	Customer cannot logon to website due to lost credentials	
Triggering Event:	Customer selects lost account information option on website	
Webpage Mock-up:	410CipherN_Lost_Account_Information.png	
Brief Description:	When a customer enters their logon credentials they are vaildated against the database. If authentication is successful, a welcome page is displayed; otherwise, options to re-enter their credentials or obtain their lost account information are presented.	
Actors:	Customer	
Related Use Cases:	Relies on: *Register new user*	
Stakeholders:	Administrators: to monitor site security	
Preconditions:	The customer account must exist and have been activated.	
Postconditions:	N/A	
Flow of Events:	Actor	System
	1. Customer enters incorrect credentials 3 times. 2. Customer enters email address and answer to password question. 3. Customer select reset link in email	1.1 Display lost account information page. 2.1 Validate password answer. 2.2 Email reset link to registered email account. 3.1 Display fields allowing use to set new password.
Exception Conditions:	2.1 If password answer is incorrect, then present error dialog to customer. 2.1 If data is not valid or all required fields are not present, then present error dialog to customer.	

Figure 27 - Lost Account Information

Use Case Name:	Cancel account	
Scenario:	Customer cancel account	
Triggering Event:	Customer selects the "cancel account" hyperlink under the Account Summary section of the website	
Webpage Mock-up:	570CipherN_Admin_Modify_Account_Status.png	
Brief Description:	When a customer selects the "cancel account" hyperlink on the website, a dialog is presented requesting additional confirmation. If the customer cortinues the account and it's associated messages are deleted.	
Actors:	Customer	
Related Use Cases:	Relies on: *Authenticate user*	
Stakeholders:	Sales department: to collect customer statistics	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case.	
Postconditions:	Confirmation email must be sent to customer.	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "profile" menu. 2. Customer selects the "cancel account" hyperlink under the Account Summary section. 3. Customer chooses to continue with cancellation.	1.1 Display current profile and account summary information. 2.1 Display a dialog requesting additional confirmation. 3.1 Display confirmation. 3.2 Disable account. 3.3 Send notification email to customer.
Exception Conditions:	3.1 If customer chooses not to cancel, then return to display from (1.1).	

Figure 28 - Cancel Account

Use Case Name:	Decrypt message (partner)	
Scenario:	Partner decrypts a message sent by another partner	
Triggering Event:	Partner downloads encrypted message	
Webpage Mock-up:	560CipherN_Admin_Retreve_Messages.png	
UML Artifacts:	CipherNSequenceDiagrams.vsd - DecryptPartnerMessage	
Brief Description:	The receiving partner logs onto the website and views an encrypted message send from the sending partner. The message attachment(s) is decrypted and saved back to the receiving partner's system.	
Actors:	Partner	
Related Use Cases:	Relies on: *Authenticate user* Relies on: *Encrypt new message*	
Stakeholders:	Customer (message sender)	
Preconditions:	Partner has successfully authenticated to the web site by invoking the *Authenticate user* use case. Partner sent message to another partner by invoking the *Encrypt new message* use case.	
Postcorditions:	Encryption key associated with message is invalidated once all recipients have retrieved the message. Email confirming delivery must be sent to message sender.	
Flow of Events:	Actor	System
	1. Partner logs in to website and seects the "messages\receive" menu. 2. Partner selects the received message of interest. 3. Partner selects browse button and selects location to download attachments.	1.1 Display an "upload" button. 2.1 Display message preview 3.1 Decrypt message. 3.2 Display progress. 3.3 Save decrypted message to selected directory. 3.4 Send delivery confirmation email to partner (message sender). 3.5 Evaluate need to invalidate encryption key based on number of remaining message recipients.
Exception Conditions:	2.1 If file dialog is unable to access local drive(s), then present error dialog. 3.1 If unable to decrypt message, then present error dialog. 3.3 If unable to save decrypted message to partner's system, then present error dialog.	

Figure 29 - Decrypt Message (Partner)

Use Case Name:	Decrypt message (non-partner)	
Scenario:	Recipient (non-partner) decrypts a message sent by a CipherN member	
Triggering Event:	Recipient selecting hyperlink in notification email	
Webpage Mock-up:	565CipherN_Admin_Retrieve_Messages_Nonpartner.png	
UML Artifacts:	CipherNSequenceDiagrams.vsd - DecryptNonPartnerMessage, NonPartnerNotification.pdf	
Brief Description:	The recipient selects the hyperlink in the email notification. The resultant CipherN webpage displays the sender provided question, allowing the recipient to enter the answer. If correct, the message and links for saving attachment(s) are displayed.	
Actors:	Recipient (non-partner)	
Related Use Cases:	Relies on: *Encrypt new message*	
Stakeholders:	Customer (message sender)	
Preconditions:	Partner sent message to another non-partner by invoking the *Encrypt new message* use case.	
Postconditions:	Encryption key associated with message is invalidated once all recipients have retrieved the message. Email confirming delivery must be sent to message sender.	
Flow of Events:	**Actor**	**System**
	1. Recipient selects notification hyperlink. 2. Recipient enters answer. 3. Recipient selects attachment link. 4. Recipient selects location to download attachment.	1.1 Determine message id based on encrypted data in the notification hyperlink. 1.2 Display webpage with sender provided question. 2.1 Validate answer. 2.2 Display message and link for saving attachment(s). 3.1 FileSave dialog is displayed. 4.1 Save decrypted attachment to selected directory. 4.2 Display save progress. 4.3 Send delivery confirmation email to partner (message sender). 4.4 Evaluate need to invalidate encryption key based on number of remaining message recipients.
Exception Conditions:	2.1 If answer is incorrect, then present error dialog. 3.1 If file dialog is unable to access local drive(s), then present error dialog. 4.1 If unable to decrypt message, then present error dialog.	

Figure 30 - Decrypt Message (Non-partner)

Use Case Name:	Produce symmary report	
Scenario:	System generates a symmary report of Cipher-N activites for a specified time period	
Triggering Event:	Specified date and time reached	
Webpage Mock-up:	SymmaryReport.pdf	
Brief Description:	When a predefined time is reached the system generates and emails a summary report of Cipher-N activities.	
Actors:	System	
Related Use Cases:		
Stakeholders:	Administrator, Customer	
Preconditions:	Customer provided an email in their profile by invoking the *Register new user* use case. Reporting task scheduled in system.	
Postconditions:	Emails containing symmary reports must be sent.	
Flow of Events:	System	
	1. System clock reaches defined date and time 2. Query database for messages sent and received during the report time period. 3. Generate individual reports specific to each customer. 4. E mail each customer their respective report to the email listed in their profile. 5. Generate complete report for all customers. 6. E mail complete report to administrator.	
Exception Conditions:	3.1 If unable to generate report, then alert administrator. 5.1 If unable to generate report then alert administrator.	

Figure 31 - Produce Summary Report

Use Case Name:	Charge customer account	
Scenario:	System bills customer, generates and emails a receipt	
Triggering Event:	Specified date and time reached	
Webpage Mock-up:	N/A	
Brief Description:	When the paid service time period for a customer is reached, the system bills the customer using the payment information on file and emails a receipt.	
Actors:	System	
Related Use Cases:	Relies on: *Upgrade account service level* Relies on: *Register new user*	
Stakeholders:	Administrator, Customer	
Preconditions:	Customer registered for (or upgraded to) a paying service level account and provided an email in their profile by invoking the *Register new user* use case. Billing task scheduled in system. Payment processing service has be contracted.	
Postconditions:	Email containing receipt must be sent. Expiration date for account entry must be updated.	
Flow of Events:	System	
	1. Billing daemon identifies account reaching expiration. 2. Query database for payment information. 3. Perform billing transaction based on payment type. 4. Update account expiration date in database. 5. Generate receipt for customer. 6. Email receipt to customer.	
Exception Conditions:	3.1 If unable to complete billing transaction, then alert customer and administrator.	

Figure 32 - Charge Customer Account

Use Case Name:	Modify customer account	
Scenario:	Administrator modifies a customer account	
Triggering Event:	Administrator selects a customer's profile on the website	
Webpage Mock-up:	580CipherN_Admin_Modify_Account.png	
Brief Description:	When an administrator selects the "profiles" menu on the website, a list of available profiles is presented. The administrator selects the profile of interest and the account's profile fields, with current data, are presented for editing.	
Actors:	Administrator	
Related Use Cases:	Relies on: *Authenticate user*	
Stakeholders:	N/A	
Preconditions:	Administrator has successfully authenticated to the website by invoking the *Authenticate user* use case.	
Postconditions:	N/A	
Flow of Events:	Actor	System
	1. Administrator logs in to website and selects the "profiles" menu. 2. Administrator selects profile of interest. 3. Administrator updates editable fields. 4. Administrator selects "enter" button to save changes.	1.1 Display list of available profiles. 2.1 Display current profile and account summary information. 4.1 Validate data and presence of required fields.
Exception Conditions:	4.1 If data is not valid or all required fields are not present, then present error dialog.	

Figure 33 - Modify Customer Account

Use Case Name:	Delete account	
Scenario:	Customer deletes account	
Triggering Event:	Customer selects the "delete account" hyperlink under the Account Summary section of the website	
Webpage Mock-up:	570CipherN_Admin_Change_Account_Status.png	
Brief Description:	When a customer selects the "delete account" hyperlink on the website, a dialog is presented requesting additional confirmation. If the customer continues the account and it's associated messages are deleted.	
Actors:	Customer	
Related Use Cases:	Relies on: *Authenticate user*	
Stakeholders:	Sales department: to collect customer statistics	
Preconditions:	Customer has successfully authenticated to the website by invoking the *Authenticate user* use case.	
Postconditions:	Confirmation email must be sent to customer.	
Flow of Events:	Actor	System
	1. Customer logs in to website and selects the "profile" menu. 2. Customer selects the "delete account" hyperlink under the Account Summary section. 3. Customer chooses to continue with deletion.	1.1 Display current profile and account summary information. 2.1 Display a dialog requesting additional confirmation. 3.1 Display confirmation. 3.2 Delete account and pending messages. 3.3 Send confirmation email customer.
Exception Conditions:	3.1 If customer select chooses not to delete, then return to display from (1.1).	

Figure 34 - Delete Account

Appendix C

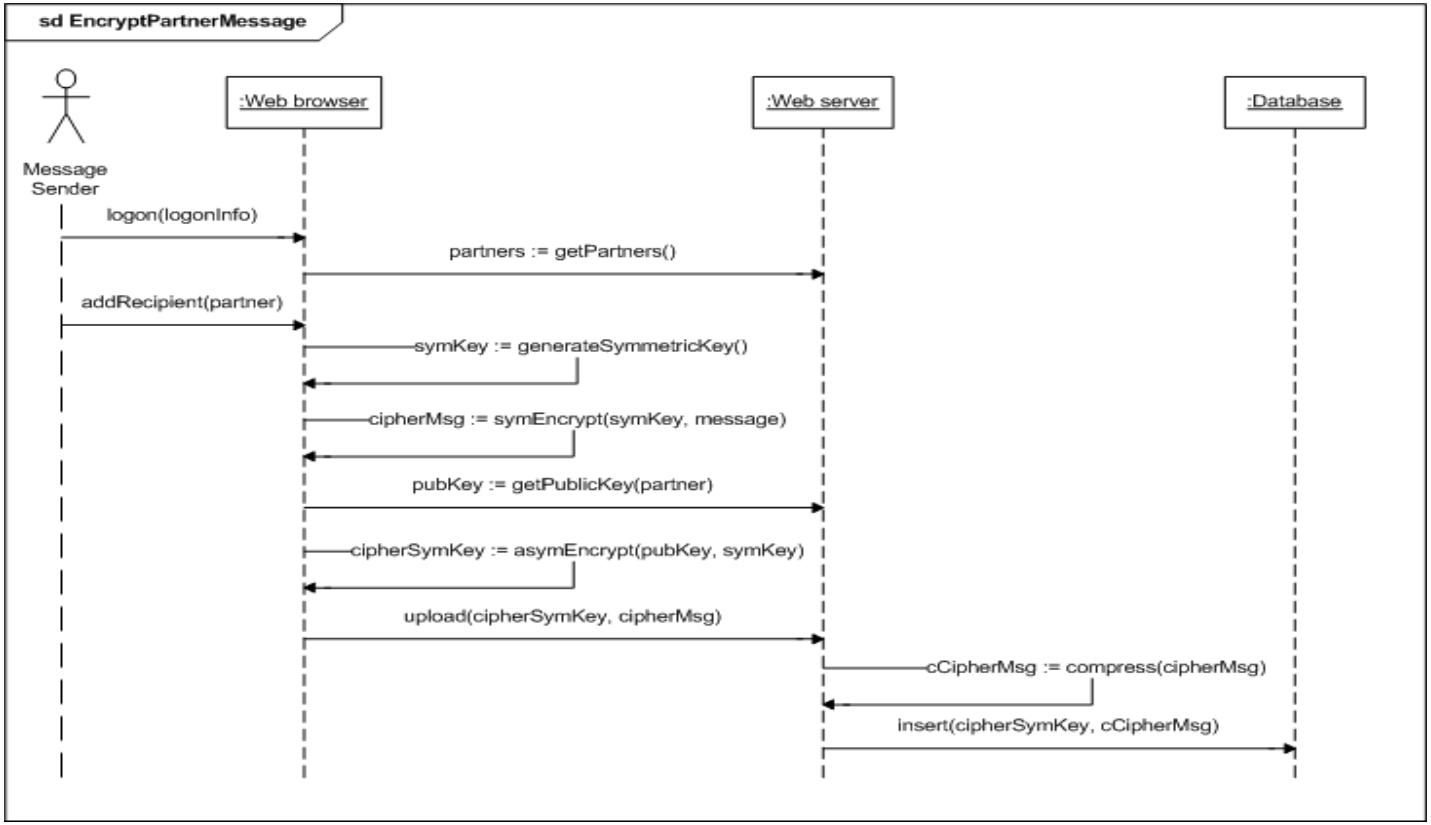

Figure 35 – Encrypt Partner Message

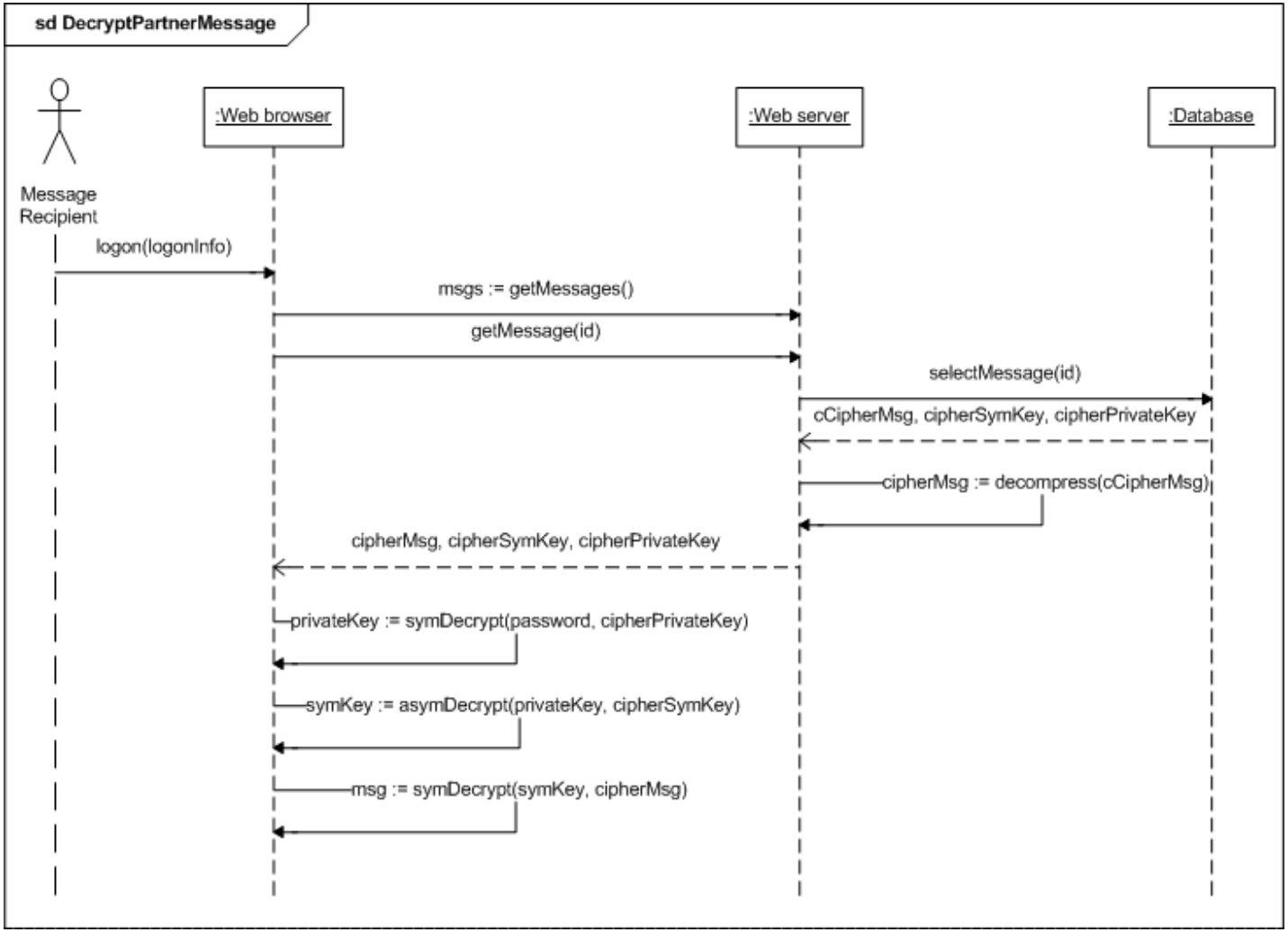

Figure 36 - Decrypt Partner Message

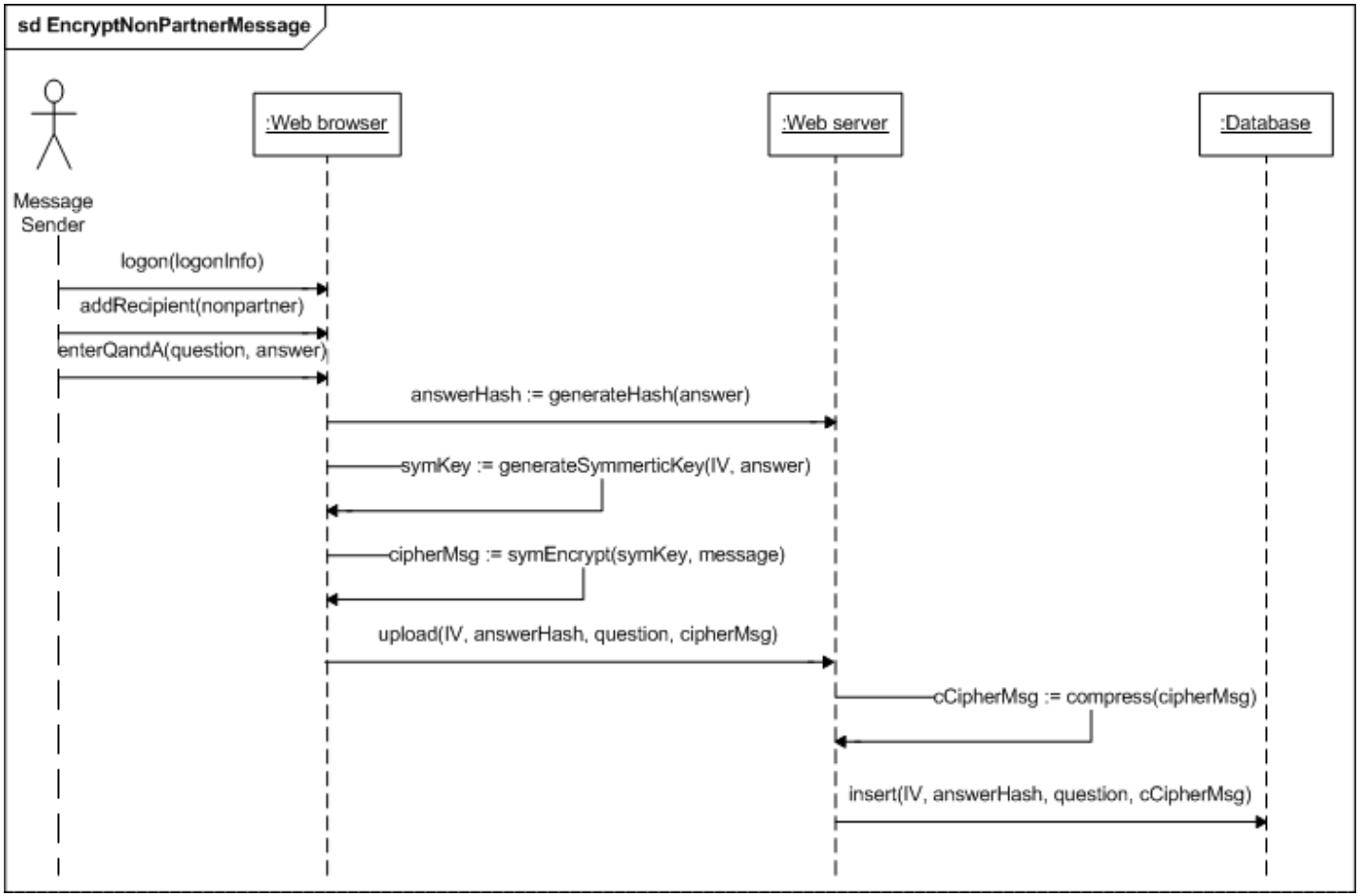

Figure 37 – Encrypt Non-Partner Message

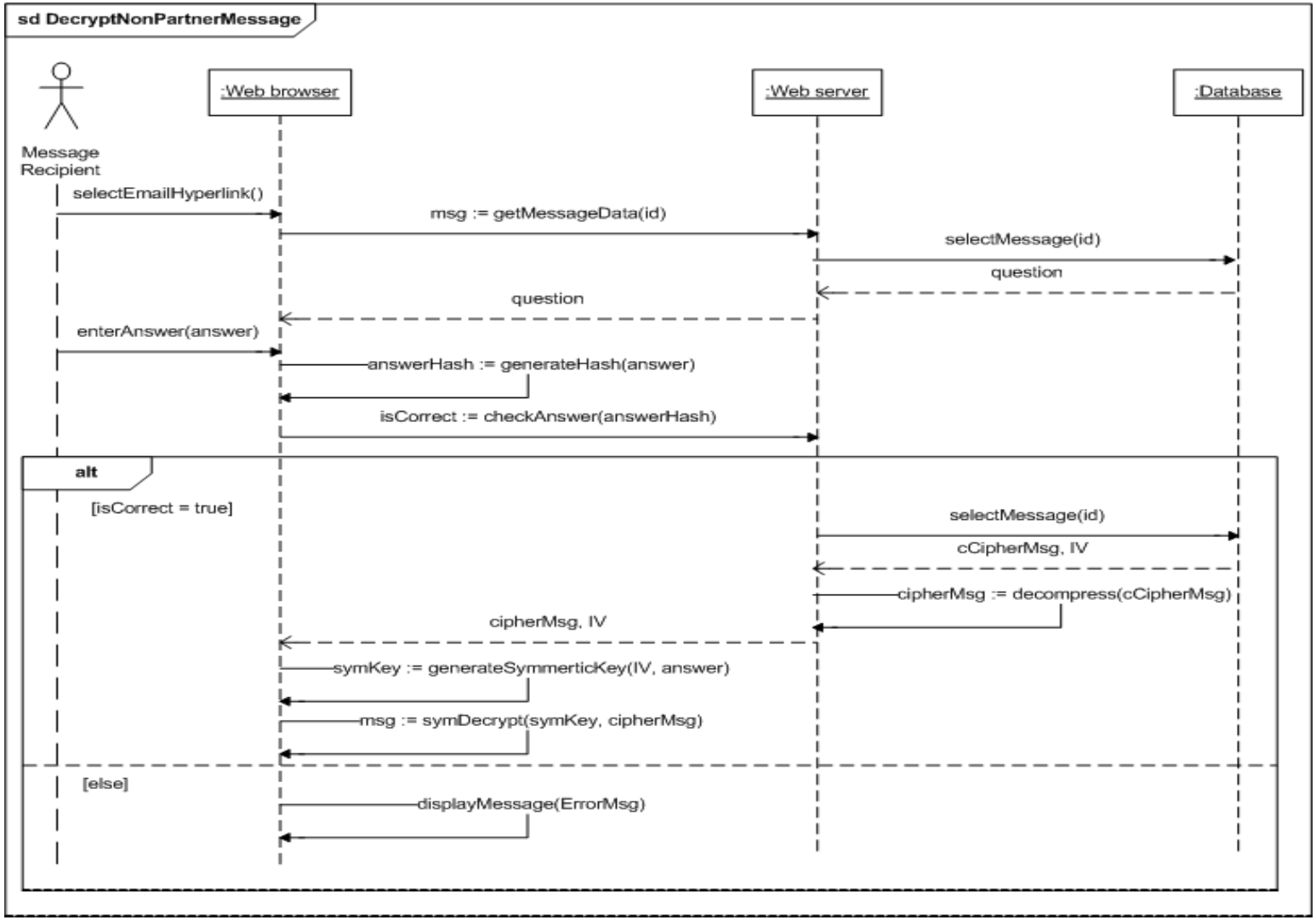

Figure 38 - Decrypt Non-Partner Message

Appendix D

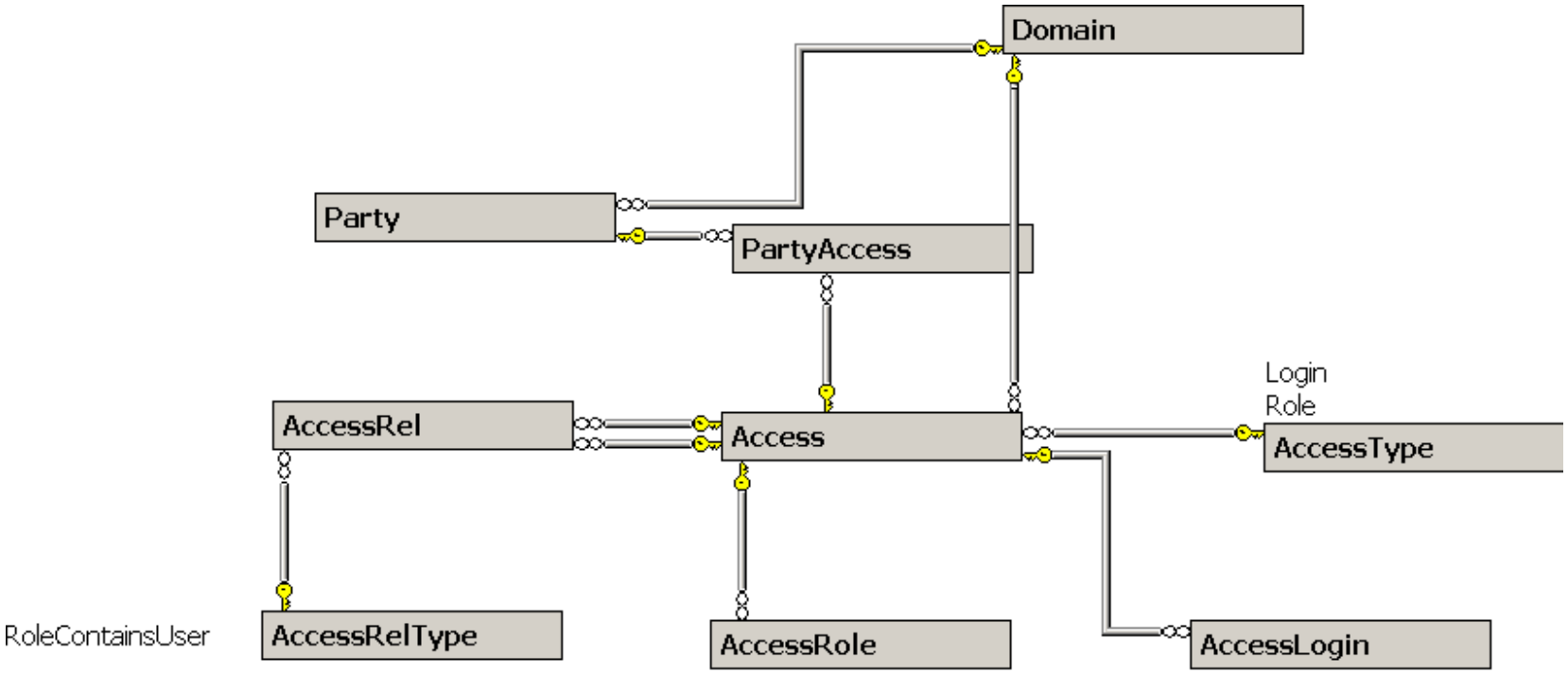

Figure 39 - Access Entity

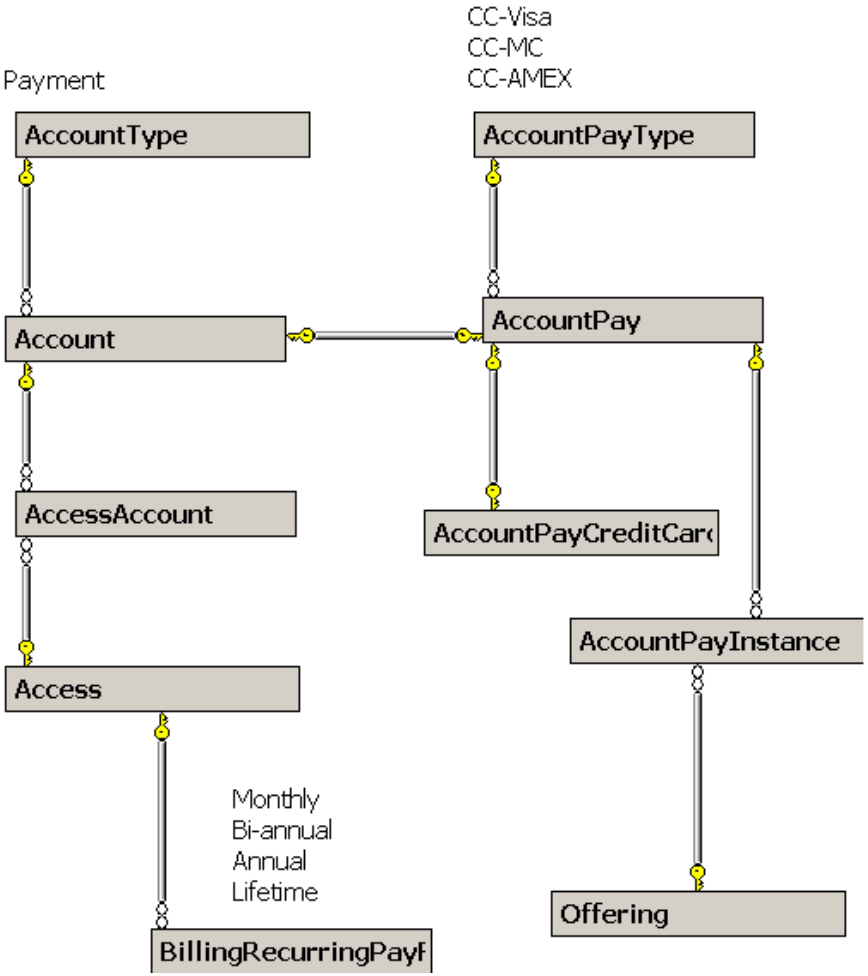

Figure 40 - Account Entity

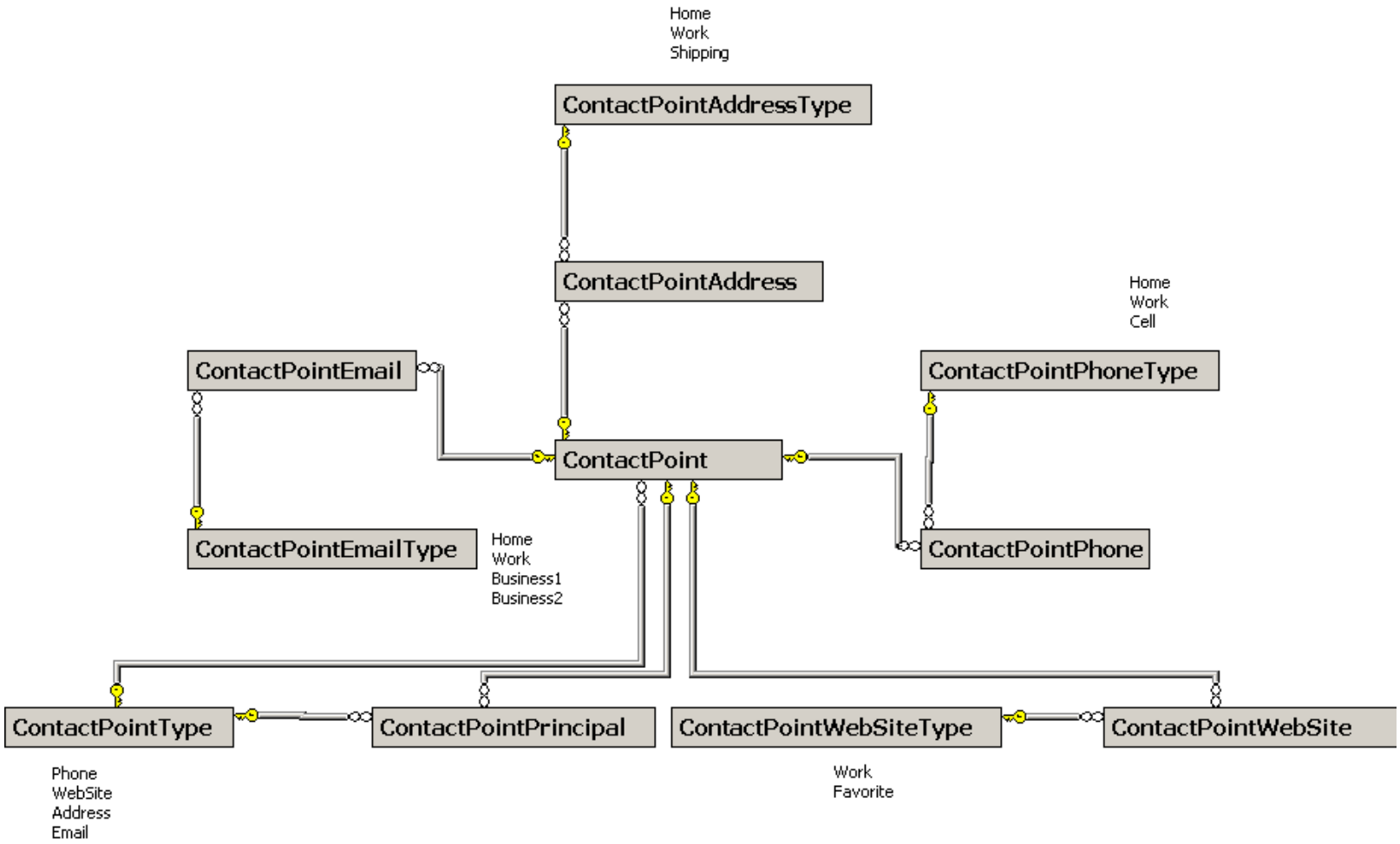

Figure 41 - ContactPoint Entity

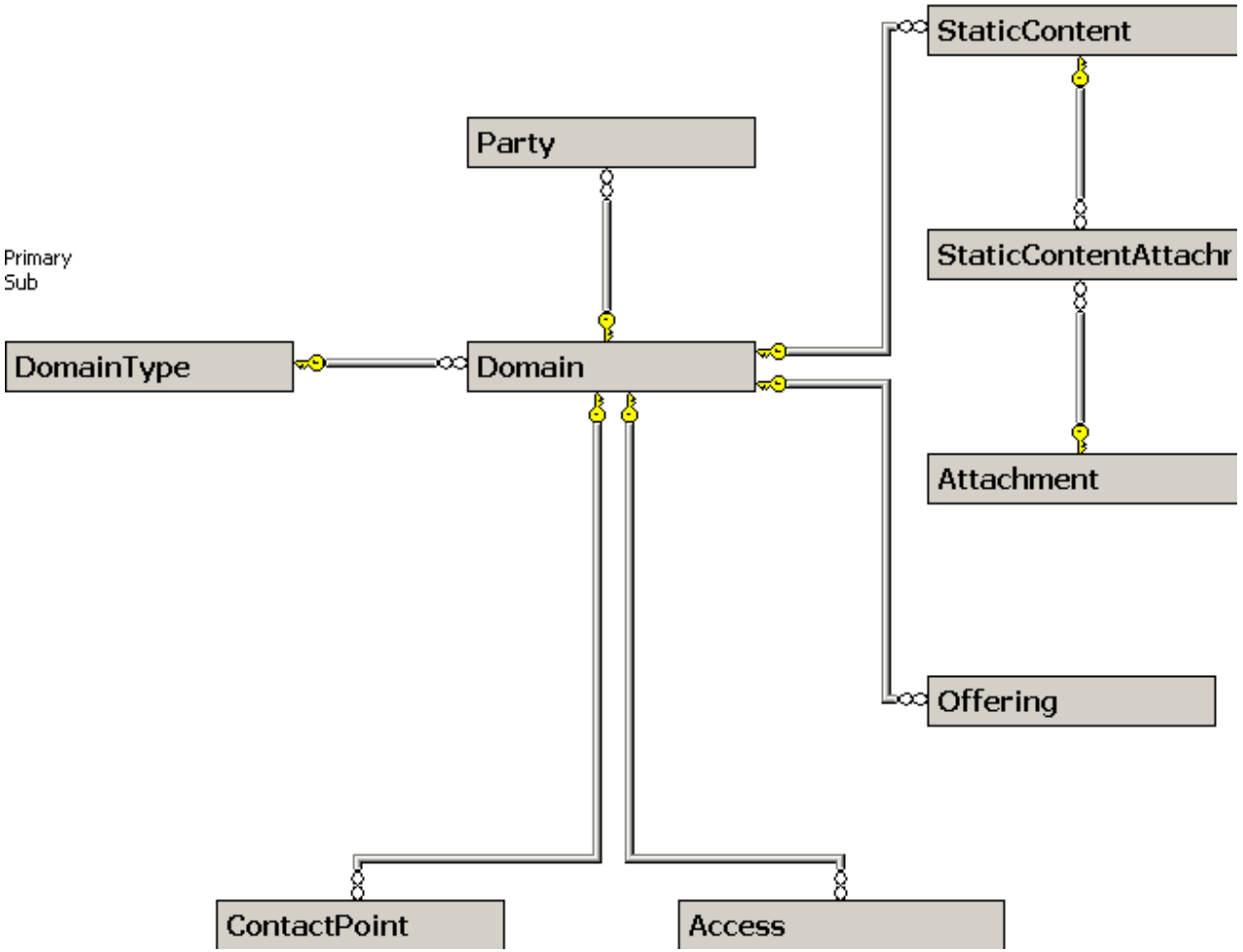

Figure 42 - Domain Entity

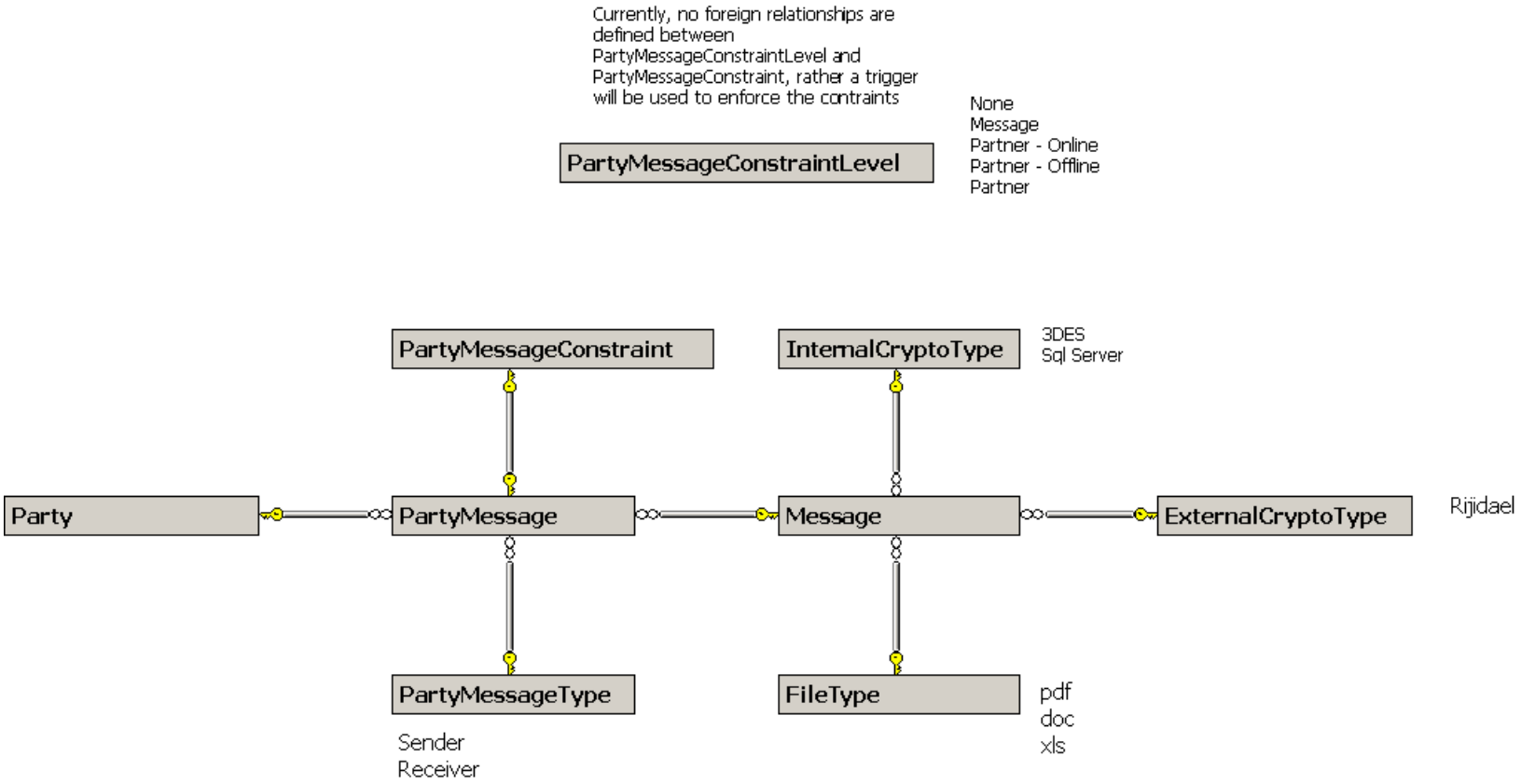

Figure 43 - Message Entity

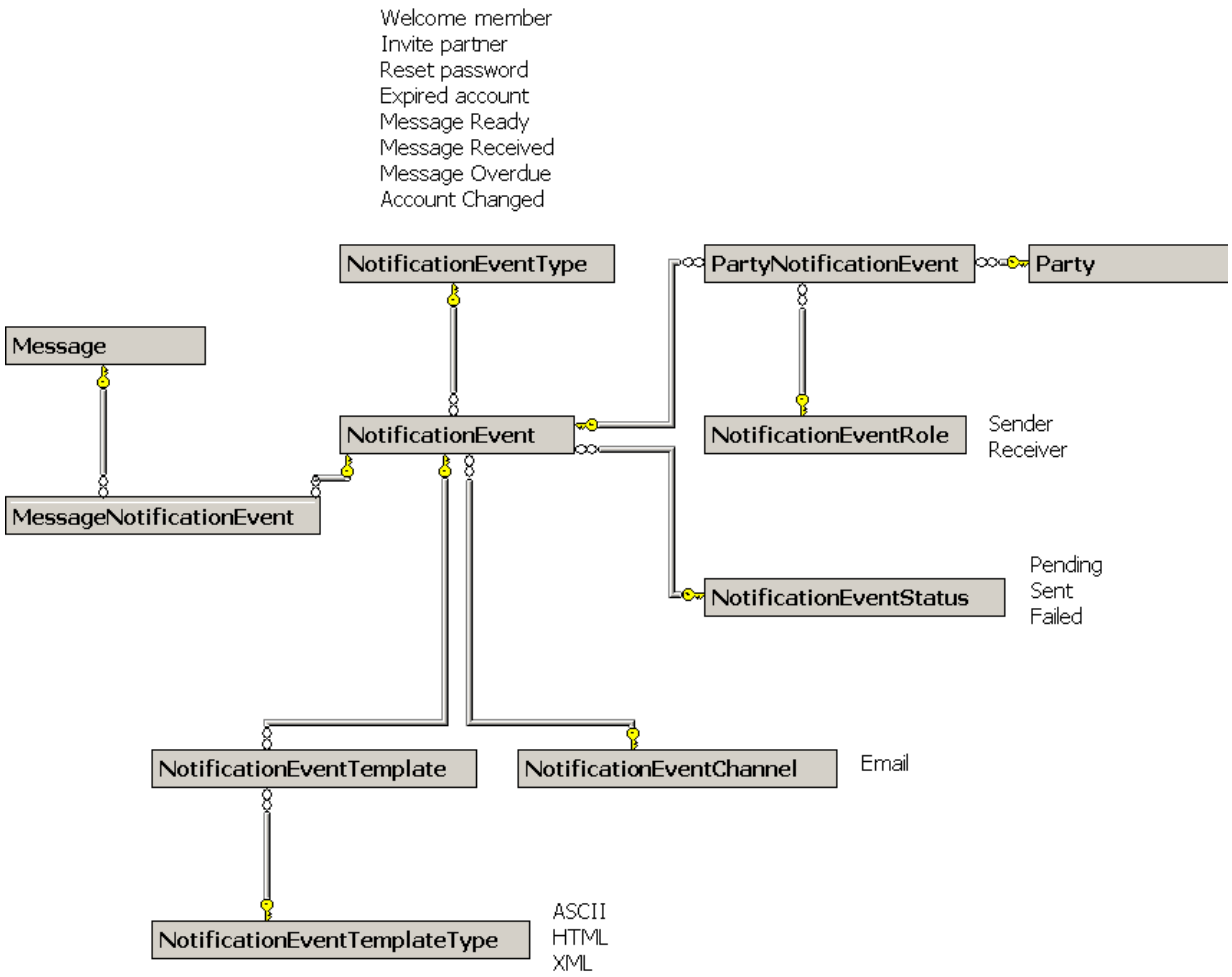

Figure 44 - Notification Entity

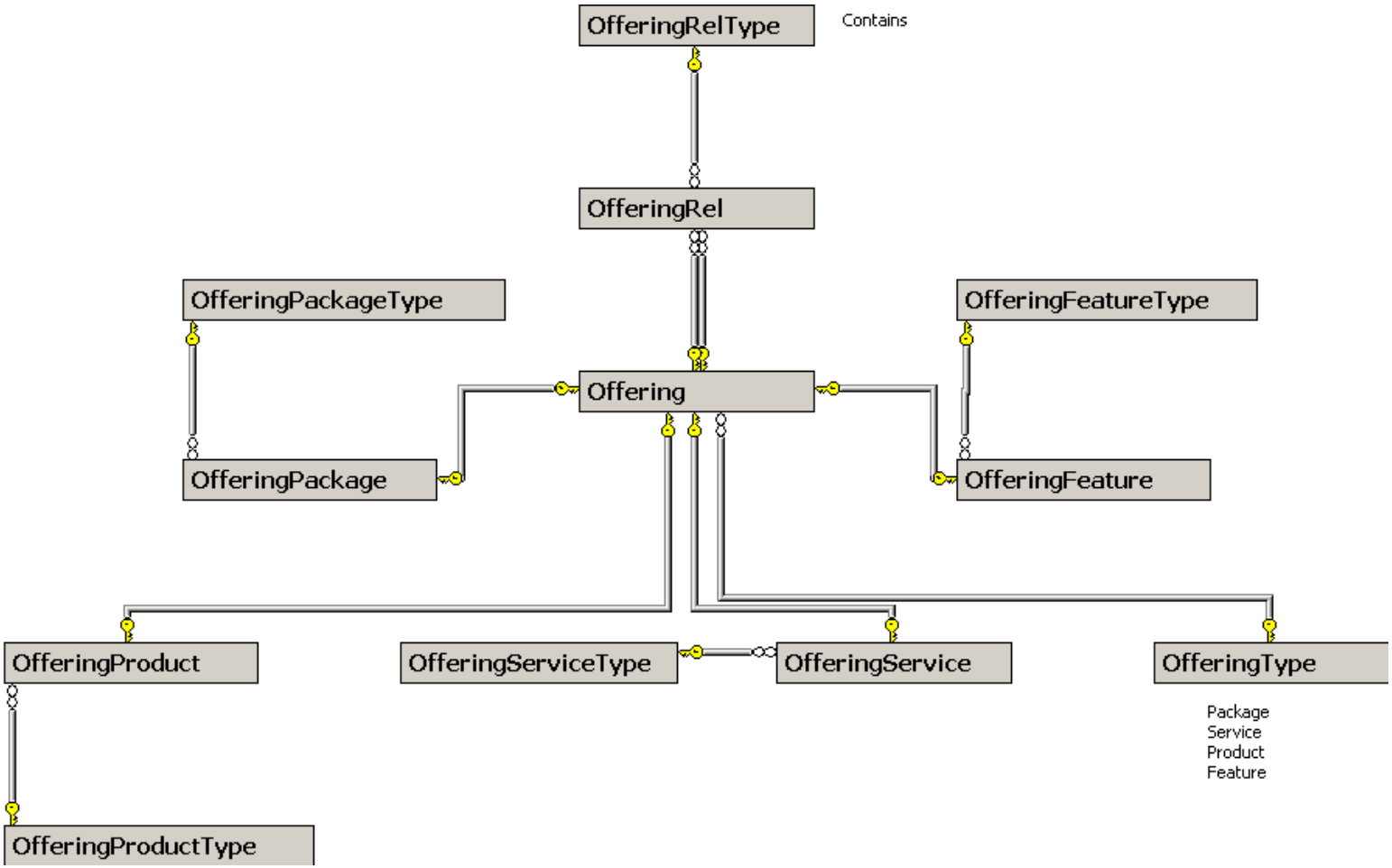

Figure 45 - Offering

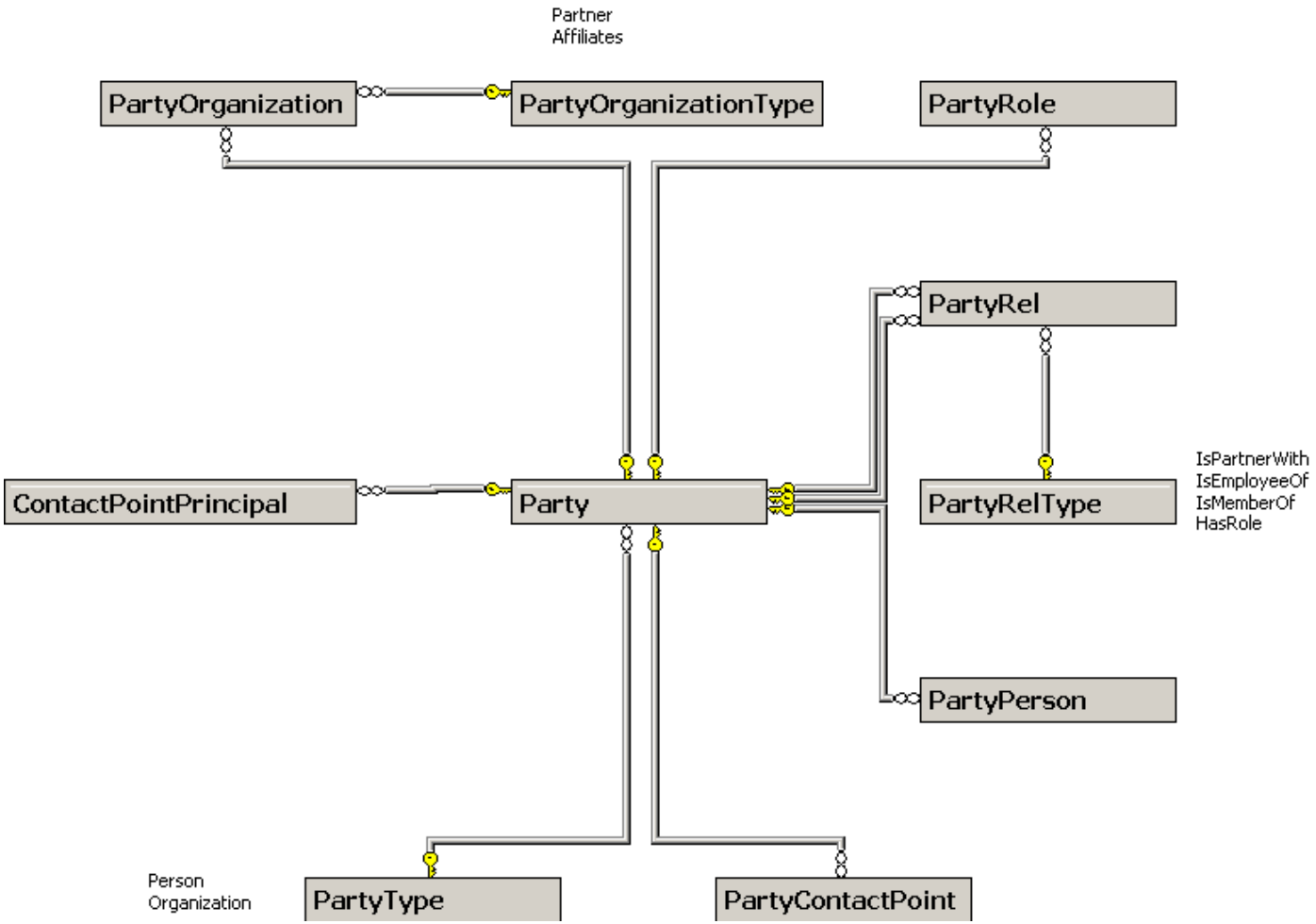

Figure 46 - Party