Universidad de Buenos Aires
Facultad de Ingeniería
Departamento de Informática

Organización de Datos (75.06)

Voto Electrónico
Documentación General

Cuatrimestre y año: 2do Cuatrimestre 2011

Docente a cargo del TP: Nicolás Pablo Fernández Theillet

Grupo: Lamas

Fecha de Entrega: 2011-11-26

Integrantes:

Padrón	***Nombre***	***Email***
91187	Gonzalez Durand, Juan Manuel	jmanuel.gonzalez.durand@gmail.com
90762	Ostrowsky, Gabriel	gaby.ostro@gmail.com
90728	Schenkelman, Damián	damian.schenkelman@gmail.com
91045	Torrado, Alejandro	aletorrado@gmail.com
90884	Zamudio, Gonzalo	ahogadosderazon@gmail.com

Índice

Especificación de Clases

La siguiente sección provee la documentación de las clases utilizadas en la solución implementada. En la misma, se detallan el nombre de la clase, la carpeta en la que se encuentra y una breve explicación del rol de la clase.

Nota: Esta sección no considera a las clases utilizadas para pruebas.

Carpeta: BPlusTree

En esta carpeta se guardan las clases que están específicamente relacionadas con el funcionamiento del árbol B+.

Ver Diagrama de Clases

Clase	Especificación
FreeBlockManager	Es utilizada para mantener actualizada la lista de bloques libres en un archivo separado al de datos. La misma utiliza un conteo incremental, siempre que no haya bloques que hayan sido liberados, en cuyo caso recurre a ellos primero.
IndexTreeBlock	Hereda de TreeBlock. Representa un bloque de la zona índice del árbol B+. Es usada por InternalNode. Guarda punteros a nodos hijo e índices (usando claves) hacia ellos.
InternalNode	Hereda de Node. Representa un nodo interno del árbol e implementa las operaciones de lectura, alta, baja y modificación de forma recursiva (llama a la función para el hijo determinado según el valor de la clave).
LeafNode	Hereda de Node. Representa un nodo hoja del árbol y todas las operaciones de lectura o escritura finalmente actúan sobre ellos.
Node	Representa, de forma abstracta, un nodo del árbol. Provee una interfaz común para operaciones como alta, baja, modificación, lectura y agrupa funcionalidad común a nodos hoja e internos.
OverflowParameter	Es utilizada para guardar información cuando un bloque entra en overflow. Se pasa como parámetro auxiliar a la función de inserción, para que los padres tengan información para actualizarse correctamente ante el split de sus hijos.
SequenceTreeBlock	Hereda de TreeBlock. Representa un bloque de la zona índice del árbol B+. Es usada por Leaf. Guarda los registros de datos de longitud variable y un puntero al nodo siguiente, para permitir el recorrido secuencial.
Tree	Agrupa la funcionalidad del árbol. De esta forma, se abstrae al usuario de la implementación a niveles más bajos del árbol y se provee una interfaz de alto nivel la cual es fácil de utilizar. Guarda un puntero a la raíz, y se encarga de guardar el último bloque leído.
TreeBlock	Hereda de BaseVariableBlock. Representa, de forma abstracta, un bloque del árbol. Provee una interfaz común para operaciones como alta, baja, modificación, lectura y agrupa funcionalidad común a bloques de índice y secuencia.
TreeBlockFile	Hereda de BaseVariableBlockFile. Es utilizada por Tree y los nodos para cargar y guardar bloques de longitud variable al archivo. Adicionalmente, al momento de descender por el árbol para las distintas operaciones, y a fin de mantener los bloques del actual recorrido en memoria, provee una pila para ir acumulando a los mismos en orden.

Carpeta: Entities

En esta carpeta se guardan las clases que están específicamente relacionadas con el funcionamiento del árbol B+.

Clase	Especificación
Administrator	Representa a un administrador del sistema. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la

	interacción con los archivos de datos.
AdministratorMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar administradores en las organizaciones de datos implementadas.
Candidate	Representa a un candidato. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
CandidateMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar candidatos en las organizaciones de datos implementadas.
Charge	Representa a un cargo. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
ChargeMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar cargos en las organizaciones de datos implementadas.
Count	Representa a un conteo. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
CountMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar conteos en las organizaciones de datos implementadas.
District	Representa a un distrito. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
DistrictMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar distritos en las organizaciones de datos implementadas.
Election	Representa a una elección. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
ElectionMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar elecciones en las organizaciones de datos implementadas.
ElectionList	Representa a una lista. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
ElectionListMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar listas en las organizaciones de datos implementadas.
Voter	Representa a un votante. Guarda sus atributos y tiene métodos para obtener/asignar los bytes. Estos métodos son usados para facilitar la interacción con los archivos de datos.
VoterMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para poder guardar votantes en las organizaciones de datos implementadas.

Carpeta: Hash

En esta carpeta se guardan las clases que están específicamente relacionadas con el funcionamiento del Hash.

Ver Diagrama de Clases

Clase	Especificación
ChargeHashingFunction	Hereda de HashingFunction. Implementa la función de hash a ser aplicada para cargos.
HashBlock	Hereda de BaseVariableBlock. Representa un bloque del Hash. Implementa los métodos para consultas, altas, bajas y modificaciones. También mantiene cuenta del espacio libre, y guarda un apuntador al bloque del archivo de

	desborde en caso de estar desbordado.
HashBlockFile	Hereda de BaseVariableBlockFile. Agrupa la funcionalidad del Hash. De esta forma, se abstrae al usuario de la implementación a niveles más bajos del Hash y se provee una interfaz de alto nivel la cual es fácil de utilizar. Además de manejar las altas, bajas y modificaciones del archivo, controla el archivo de desborde. También provee funcionalidades de más bajo nivel como carga de bloques de forma relativa.
HashingFunction	Provee una interfaz común para la función de Hash a ser utilizada. De esta forma se puede abstraer de los elementos que se están guardando sin modificar la implementación del Hash.
VoterHashingFunction	Hereda de HashingFunction. Implementa la función de hash a ser aplicada para votantes.

Carpeta: Helpers

Esta carpeta contiene clases con métodos auxiliares (para ser usados de forma estática).

Clase	Especificación
ByteOperators	Tiene dos métodos: • Setear un bit en un bit • Verificar si un bit dentro de un byte vale 1.

Carpeta: Indexes

En esta carpeta se guardan las clases relacionadas con los índices adicionales a los archivos de datos requeridos para diferentes funcionalidades.

Clase	Especificación
CountId	Guarda información sobre el identificador de un Conteo. Es utilizada para indexar conteos.
DistrictCounts	Guarda una lista de conteos relacionados con un distrito. A partir de esta clase se generan los registros del índice de conteos por distrito.
DistrictCountsIndex	Provee una abstracción sobre el árbol B+ para indexar y des-indexar conteos relacionados con un distrito. Adicionalmente, permite recuperar una entrada indexada a partir de un distrito determinado.
DistrictCountsMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para DistrictCounts.
DistrictElections	Guarda una lista de elecciones relacionadas con un distrito. A partir de esta clase se generan los registros del índice de elecciones por distrito.
DistrictElectionsIndex	Provee una abstracción sobre el árbol B+ para indexar y des-indexar elecciones relacionadas con un distrito. Adicionalmente, permite recuperar una entrada indexada a partir de un distrito determinado.
DistrictElectionsMethods	Hereda de RecordMethods. Implementa los métodos de esa clase para DistrictElections.
ElectionId	Guarda información sobre el identificador de una Elección. Es utilizada para indexar conteos.

Carpeta: VariableBlocks

En esta carpeta se guardan las clases que implementan funcionalidad relacionada con todos los archivos que usan registros de longitud variable.

Ver Diagrama de Clases

Clase	Especificación
BaseVariableBlock	Representa de manera abstracta un bloque de un archivo con registros de longitud variable. Guarda el espacio libre disponible, la posición donde comienzan los registros y provee una interfaz común para algunas operaciones como altas, bajas y modificaciones de registros. Adicionalmente, implementa operaciones comunes a todos los bloques, en algunos casos en forma total y en otros como Template Method.
BaseVariableBlockFile	Provee una interfaz común para todos los archivos que usan bloques de longitud variable, así como también implementaciones base de métodos relacionados con posicionamiento relativo en el archivo.
Constants	Guarda constantes relacionadas con archivos de longitud variable. Estas son: • Tamaño del espacio libre del bloque. • Tamaño del espacio reservado para campo de control de la longitud de un registro. • Tamaño del espacio reservado para campo de control de la longitud de un campo.
RecordMethods	Provee una interfaz común que es usada por las diferentes estructuras que usan registros de longitud variable. Es una alternativa al pasaje de punteros a funciones. Permite, de forma genérica: • Comparar registros • Obtener la clave a partir de un registro de datos • Imprimir clave • Imprimir registro
SimpleVariableBlock	Hereda de BaseVariableBlock. Es el tipo de bloque utilizado por SimpleVariableBlockFile. Maneja el espacio libre encadenando listas de espacio libre que se encuentran en diferentes bloques.
SimpleVariableBlockFile	Hereda de BaseVariableBlockFile. Es una implementación de un archivo secuencial para guardar registros de longitud variable y organizado en bloques. Es usado por el Hash para guardar los bloques de desborde.
VariableRecord	Abstracción utilizada para que cualquier entidad pueda ser guardada en el árbol B+ y Hash sin necesidad de tener una implementación particular para ella. Permite guardar los registros como bytes y permite acceder a estos bytes a su tamaño.

Carpeta: Voting

Agrupa funcionalidad relacionada con la aplicación, específicamente con la funcionalidad de voto.

Clase	Especificación
Configuration	Lee el archivo de configuración donde está especificado el nombre de los archivos para guardar cada tipo de entidades, el tamaño de los bloques de los mismos, el archivo de donde cargar los datos inicialmente (e información como cantidad de bloques iniciales y tamaño promedio del bloque en el caso de archivos a ser organizados de forma directa). Expone las entradas obtenidas mediante la clase ConfigurationEntry.
ConfigurationEntry	Representa una entrada del archivo de configuración de entidades. Guarda el tamaño de bloque del archivo donde se guardara cada una, así como el archivo de datos de carga inicial, el nombre del archivo de datos, y en el caso de entidades a ser guardadas en archivos con organización directa, el tamaño

	promedio del bloque y la cantidad inicial de bloques.
DataFileLoader	Clase encargada de la carga inicial de los archivos de cada entidad a partir de archivos de texto plano con campos separados por comas.
Log	Sirve para guardar registros de operaciones realizadas en el sistema, particularmente, aquellas relacionadas con el voto.
Menu	Utilizada para la interacción con el usuario. Específicamente, sirve para mostrar menús y leer la entrada del usuario de forma consistente a lo largo de la aplicación.
Voting	Maneja el flujo de voto. Es utilizada para la carga inicial de datos, ya que hace votar a todos los electores en todas las elecciones disponibles para ellos.

Carpeta: Vigenere

En esta carpeta se guardan las clases que están específicamente relacionadas con el cifrado Vigenere, incluyendo encriptado, desencriptado y el ataque de Kasiski para romperlo.

Ver Diagrama de Clases

Clase	Especificación
VigenereCipher	Es utilizada para encriptar/desencriptar cadenas de caracteres con una clave determinada por el usuario.
Kasiski	Aquí se implementa el ataque de Kasiski para romper Vigenere. Se precisa del criptograma y de la longitud de nGrama a utilizar. Este último parámetro debe ser un número mayor a 1 pero se recomienda que no sea mayor a 4.

Carpeta: RSA

En esta carpeta se guardan las clases que están específicamente relacionadas con la encriptación RSA. Aquí se encuentra la lógica necesaria para encriptar/desencriptar archivos, proveyendo también la habilidad de romperlo conociendo solo la clave pública.

Ver Diagrama de Clases

Clase	Especificación
RSACipher	Esta clase se encarga del manejo general del algoritmo, recibe por parámetro las claves privada y pública y encripta/desencripta el mensaje/criptograma que recibe.
RSAKey	En esta clase se guarda el par (n, e) utilizado como key por *RSACipher*
RSAKeySet	Es la encargada de generar el par de claves (privada y pública) mediante la ayuda de *PrimeGenerator*.
RSAAttacker	Esta clase se encarga de romper la encriptación RSA conociendo la clave pública de dicha encriptación.
PrimeGenerator	Genera los números primeros necesarios para generar las claves privada y pública utilizadas por *RSACipher*.
KeyManager	Clase encargada de la lectura/escritura de las claves a disco

Carpeta: Menu

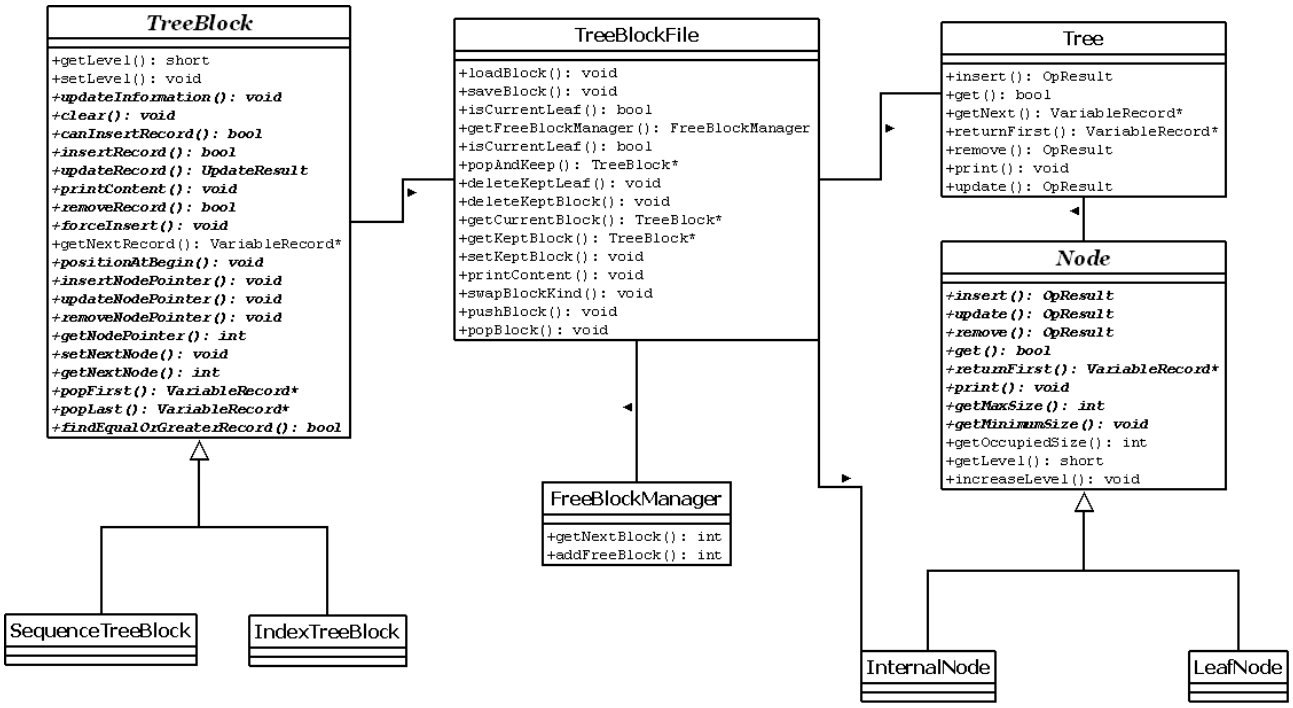
En esta carpeta se guardan las clases donde se implementó la lógica del menú.

Clase	Especificación
MainMenu	En esta clase se guarda toda la lógica relacionada con el funcionamiento del menú, de manera que se provee al usuario un menú por consola que facilita el uso del programa.

Diagramas de Clases

La siguiente sección provee diversos diagramas de clase con el fin de mostrar las relaciones estáticas entre las clases. En la misma, también se proveerá una breve explicación de cada diagrama.

Árbol B+

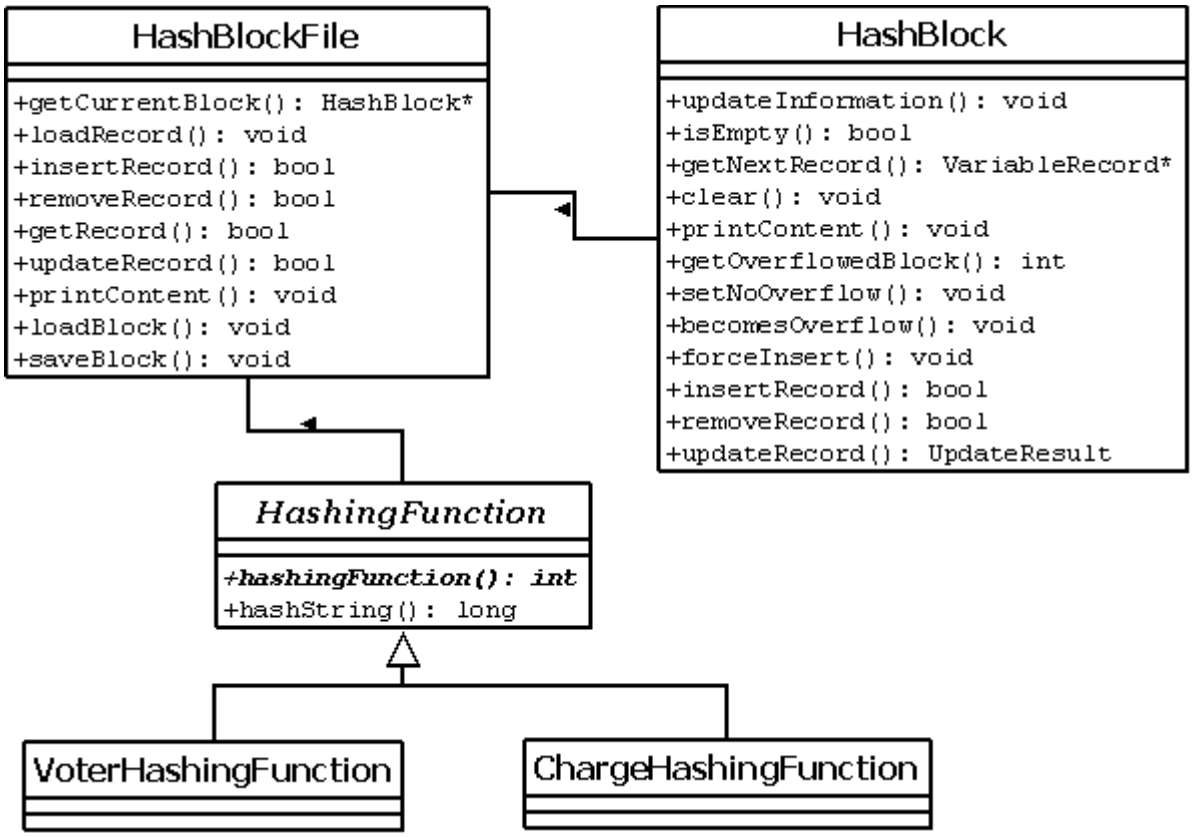


Nota: Las operaciones con fuente negrita y cursiva son virtuales, así como también los nombres de las clases en cursiva son abstractas.

Como se muestra en el diagrama, el árbol B+ (clase *Tree*) está asociada a la clase abstracta *Node* (debido a la raíz), mediante la cual ejecuta sus operaciones absteniéndose de si se trata de un nodo interno (clase *InternalNode*) o un nodo hoja (clase *LeafNode*). Por otro lado, la clase *Tree* se asocia también con la clase *TreeBlockFile*, que es la encargada de la carga y descarga de bloques en el archivo. Esta asociación permite al árbol acceder a distintos bloques dentro del archivo.

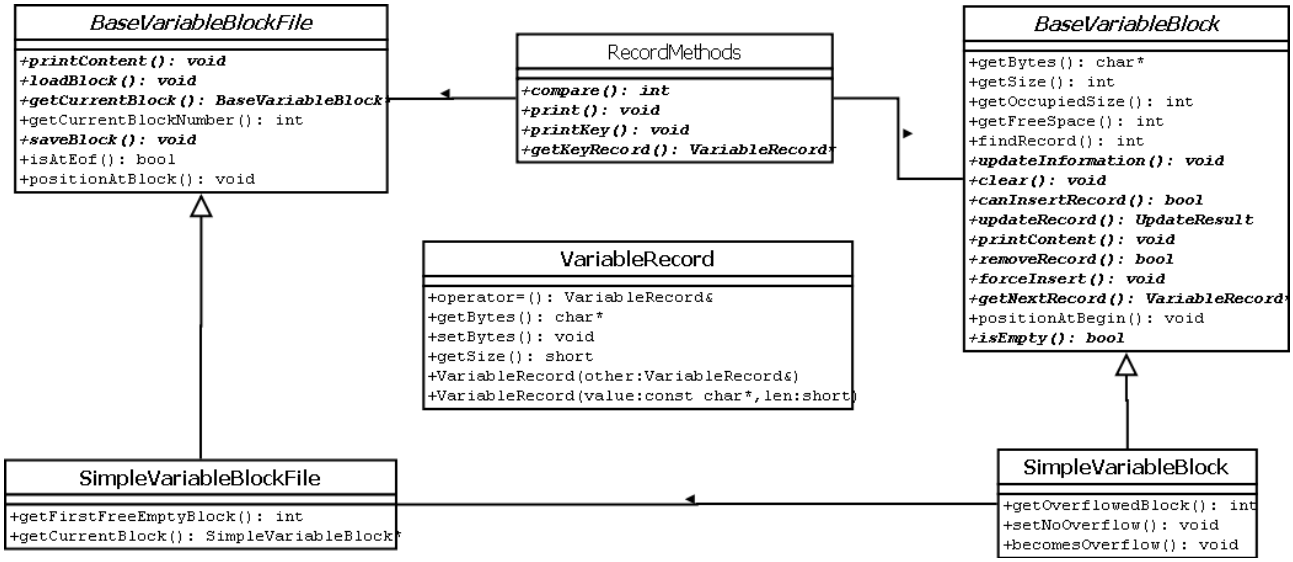
Por otro lado, la clase *TreeBlockFile* posee a la clase *FreeBlockManager* para manejar los bloques libres dentro del archivo y está asociada a la clase abstracta *TreeBlock*, que es utilizada polimórficamente para manejar los bloques. Se puede ver en el diagrama los métodos abstractos de *TreeBlock* que son implementados por *SequenceTreeBlock* e *IndexTreeBlock* según la necesidad.

Hash



Este diagrama muestra las relaciones más importantes de las clases que involucran al Hash. La clase *HashBlockFile* es en este caso la interfaz de alto nivel provista al usuario, mediante la cual se efectúan todas las operaciones. Esta clase está asociada a la clase *HashBlock* la cual se encarga de las operaciones con registros. Por otro lado, se ve la asociación entre *HashBlockFile* y la clase abstracta *HashingFunction*, actuando como interfaz de función Hash. Finalmente, la relación de herencia de las clases *VoterHashingFunction* y *ChargeHashingFunction* con *HashingFunction* que sirven para reutilizar el hash para distintas entidades.

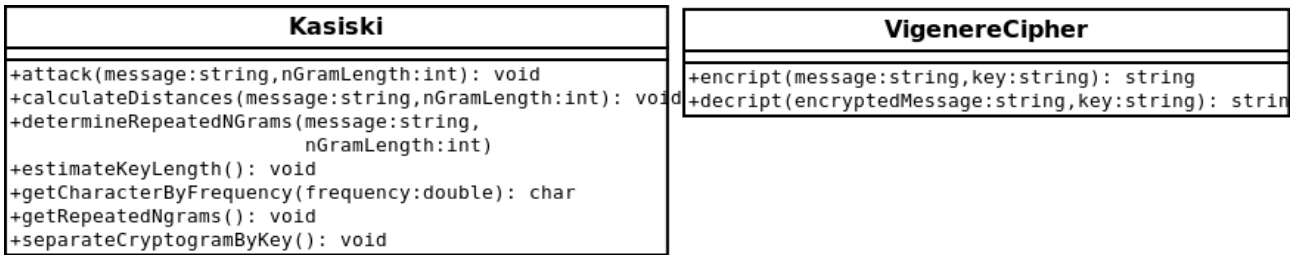
Archivos de Bloques Variables



Este diagrama da una idea de la relación que existe entre las principales clases involucradas en la carga y descarga de registros en bloques variables. Como se puede ver, la clase abstracta *BaseVariableBLockFile*

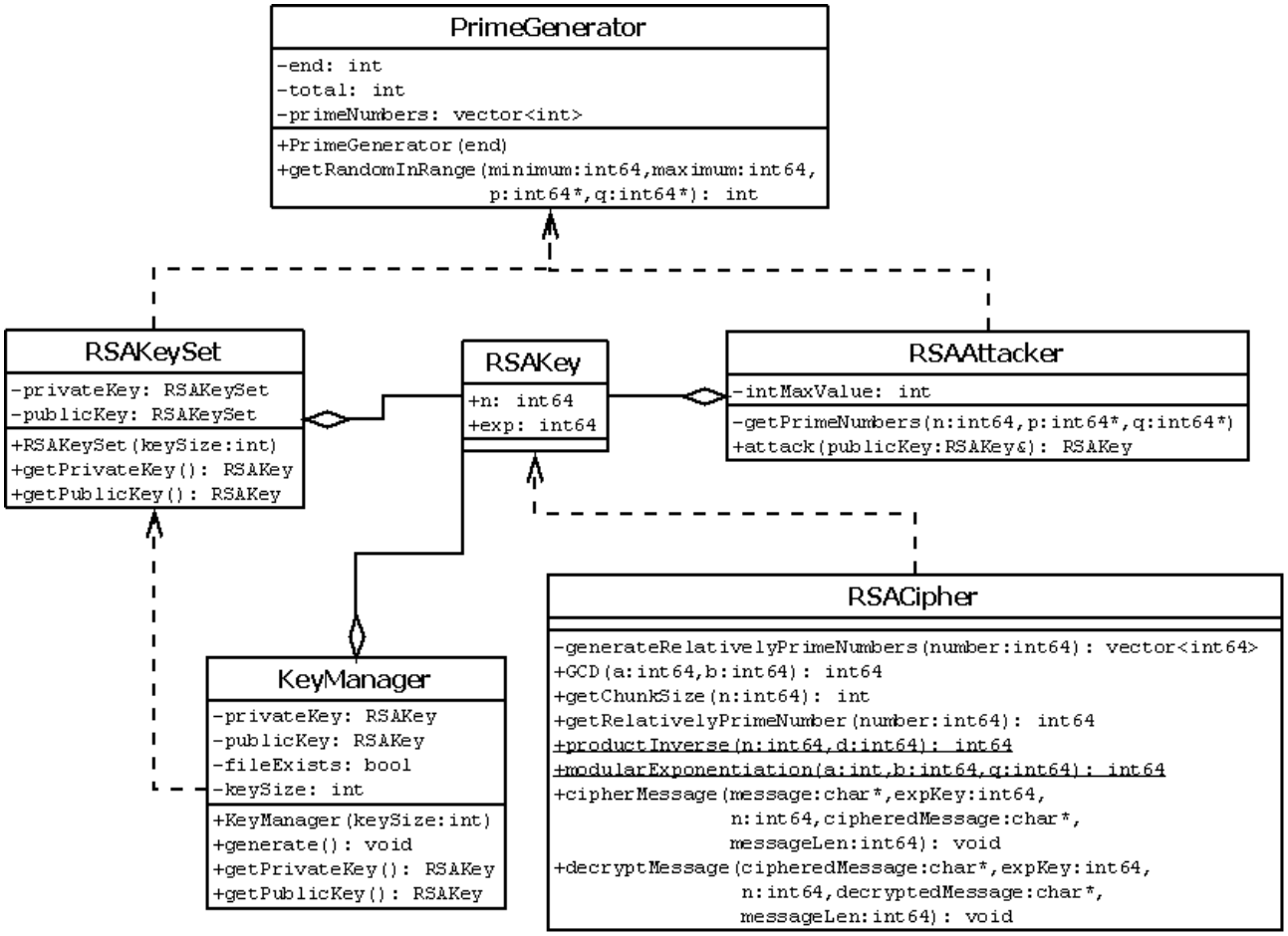
actúa de interfaz para permitirle al usuario manejar el archivo en bloques abstrayéndolo de la implementación. De dicha clase deriva la clase *SimpleVariableBlockFile* que es la encargada de administrar archivos secuenciales de bloques de longitud variable. A su vez, utiliza la clase *SimpleVariableBlock* la se encarga del manejo de registros, respetando la interfaz propuesta por la clase abstracta *BaseVariableBlock*. Vemos también que *RecordMethods* está presente en *BaseVariableBlockFile* y *BaseVariableBlock* ya que es necesaria para las operaciones entre registros lo cual permite a nuestros archivos guardar registros de cualquier tipo. Finalmente, se encuentra la clase *VariableRecord* que si bien no está asociada a ninguna clase directamente, ésta es utilizada por las clases principales del diagrama ya que es la clase que representa el registro de longitud variable.

Vigenere



Este diagrama si bien no aporta demasiado ya que no hay asociaciones en las clases, da una clara idea de que solo se utilizaron dos clases principales: una para cifrado Vigenere y otra para su correspondiente ataque, Kasiski.

RSA



En el diagrama se muestra como la clase RSACipher es la clase central, la cual recibe el par de claves pública y privada previamente generadas por RSAKeySet con ayuda de PrimeGenerator. Una vez que RSACipher tiene las claves, este encripta el mensaje que se le pasa. Por otro lado tenemos a RSAAttacker que utiliza RSAKey para guardar la clave pública necesaria para empezar el ataque y el PrimeGenerator. Finalmente, la clase KeyManager se encarga de la lectura/escritura de disco mediante la utilización de RSAKey y RSAKeySet.

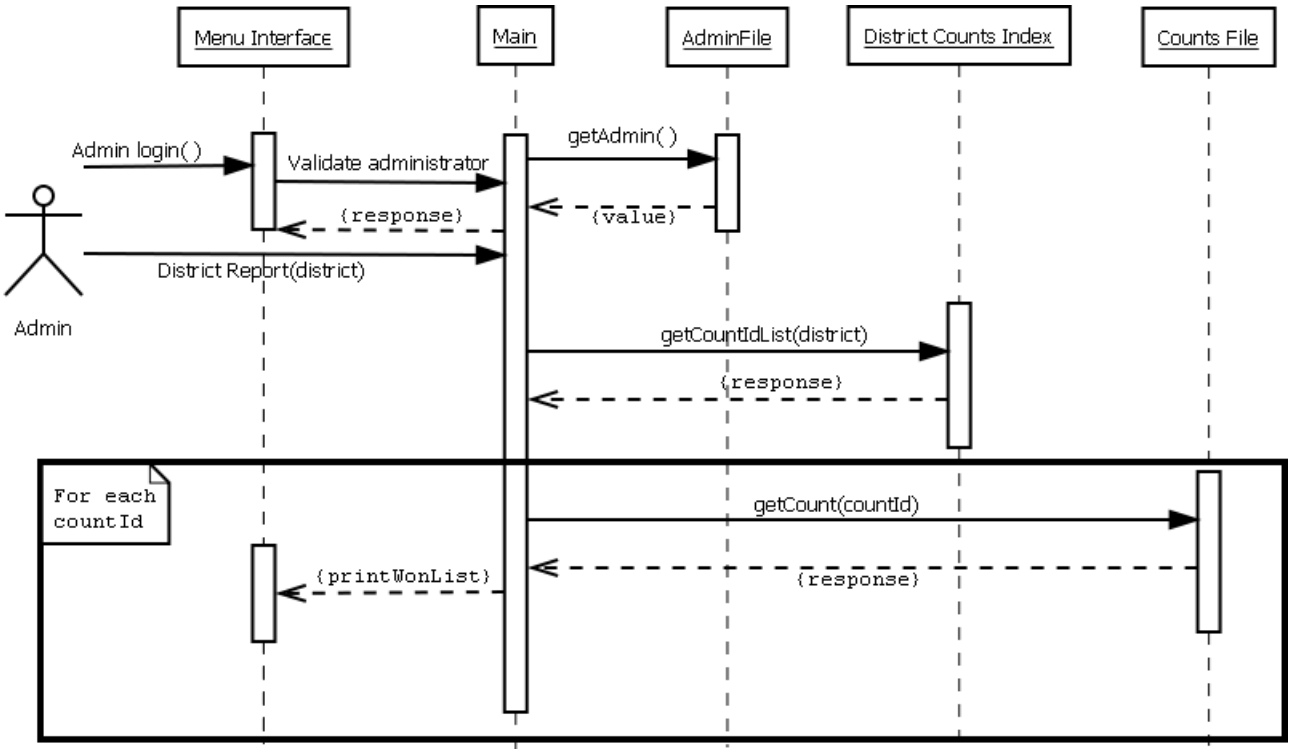
Diagramas de Secuencia

En esta sección se presentan algunos diagramas de secuencia que detallan como se implementaron las siguientes funcionalidades:

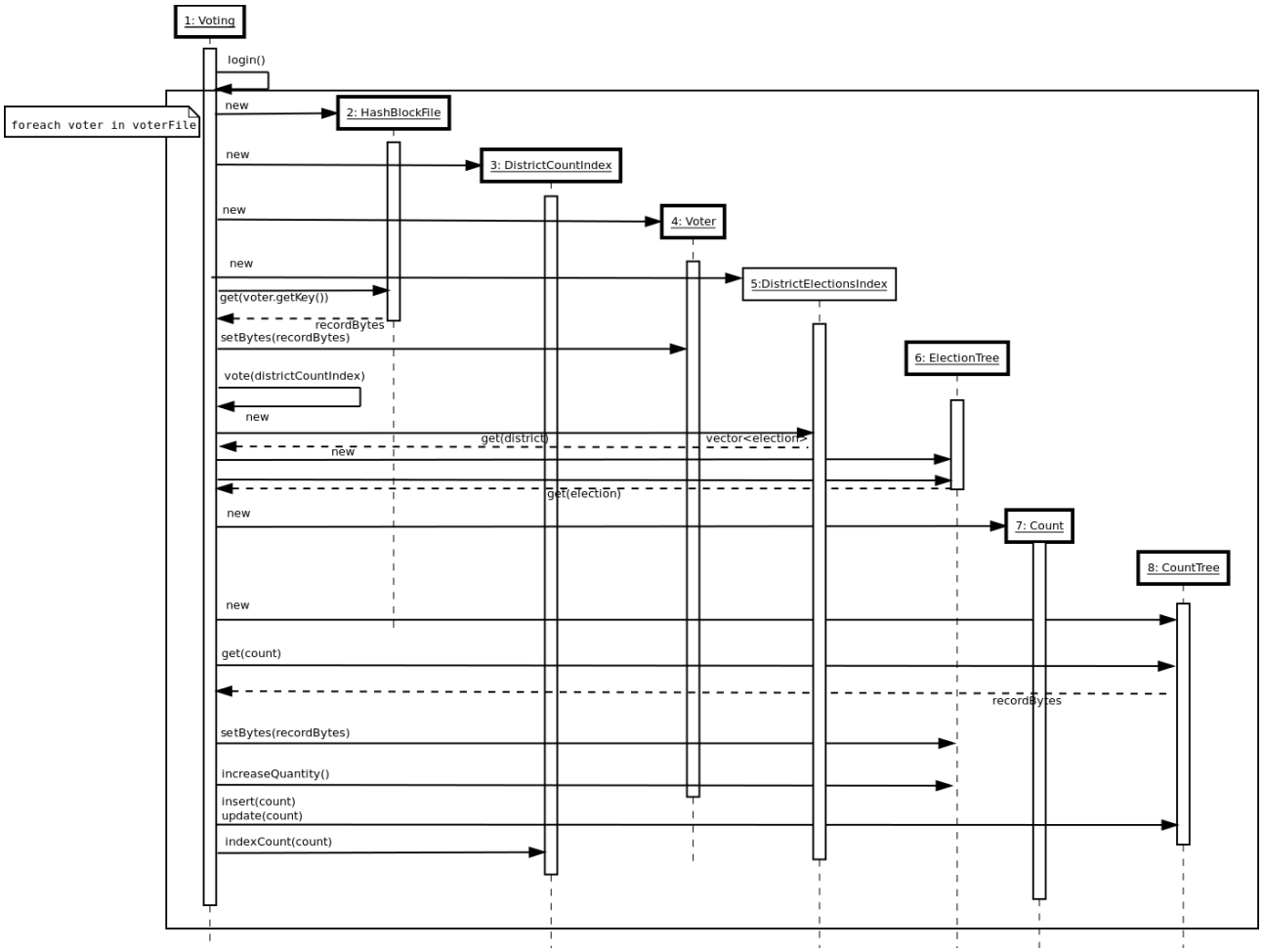
- Reporte Elecciones por Distrito
- Generación votos aleatorios

Elegimos estas secuencias porque muestran interacción con tanto el árbol B+ como el Hash asi como también el uso de índices secundarios.

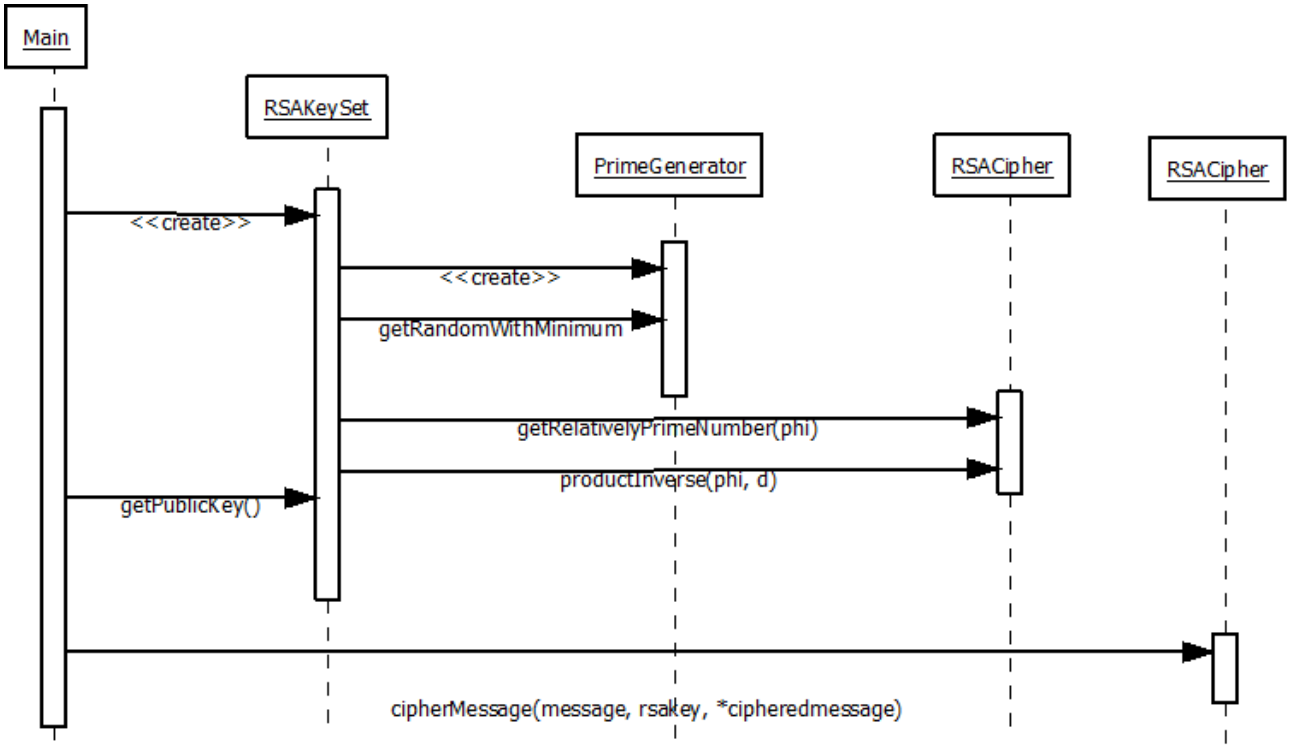
Reporte Elecciones por Distrito



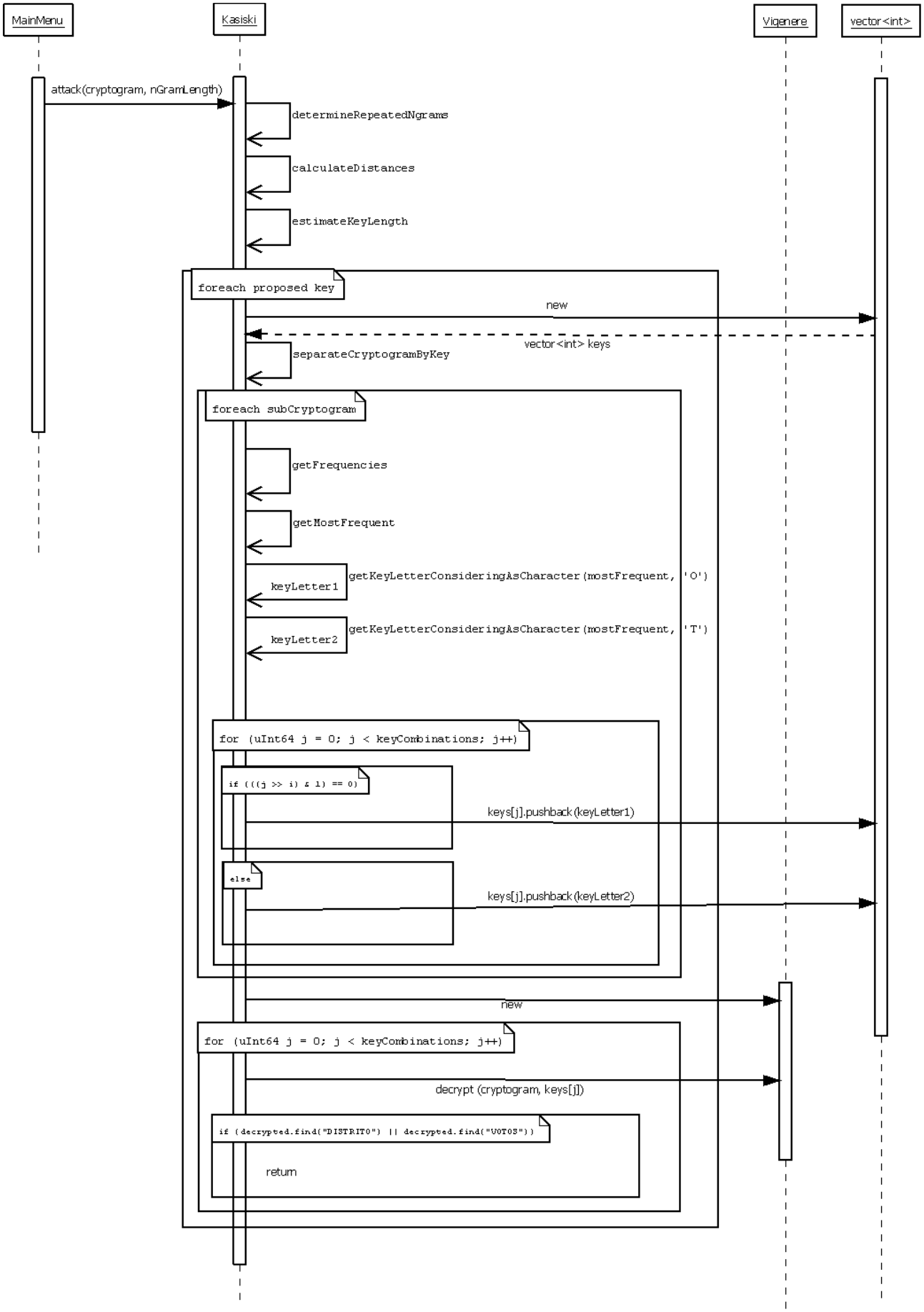
Generación Votos Aleatorios



Encriptación RSA y generación de claves



Kasiski: Ataque para Cifrado Vigenere



Known Issues

A continuación se detalla la lista Known Issues que quedaron pendientes respecto a todos los que fueron encontrados durante el desarrollo y las pruebas (ver tabla de severidades y prioridades).

Estado	Prio	Severidad	Detalle	Comentario
Pendiente	Alta	2	Validaciones para la baja de entidades	Ejemplo: Borrar una elección si hay listas asociadas a ella
Completa	Alta	3	Validación de existencia de archivos poblados	Previamente a hacer log-in, se deben poblar los archivos
Completa	Media	4	Al insertar un registro duplicado, la inserción falla sin indicar la causa	
Completa	Media	4	Validaciones para la inserción de fechas	
Completa	Baja	4	Se imprime dos veces la password al hacer log-in	

Tabla de Severidades

Severidad	Criterio de filtrado
SEV 1 (Bloqueante)	Afecta datos o funcionalidad crítica en un feature importante, afectando severamente a los usuarios, sin un workaround posible.
SEV2 (Severo)	Afecta datos o funcionalidad crítica en un feature importante, y tiene un workaround complicado. En caso de pérdida o corrupción de datos el problema es difícil de detectar por el usuario.
SEV 3 (Periférico)	Afecta datos o funcionalidad no crítica en un feature importante, y tiene un workaround fácil.
SEV 4 (Cosmético)	Afecta la estética, o que la aplicación tenga una vista profesional.

Tabla de Prioridades

Prioridad	Criterio de Filtrado
Muy Alta	Relacionados con features que afectan datos o funcionalidad crítica y son usados todo el tiempo.
Alta	Relacionados con features que afectan datos o funcionalidad crítica y son usados todo con frecuencia.
Media	Relacionados con features que son usados algunas veces.
Baja	Relacionados con features que son raramente usados.