EXHIBIT 1
HARRY DIREEN, PH.D.
EXPERT REPORT
Voltage SecureData provides software packages for encrypting and decrypting data to be protected within databases. The VibeSimple PL/SQL package is designed to work with Oracle databases, Figure 3. Voltage provides User Defined Functions (UDFs) to operate with TeraData databases, Figure 4.

---

Using the VibeSimple PL/SQL Package

Voltage provides an implementation of VibeSimple for PL/SQL. The VibeSimple functions are described in Chapter 3, "VibeSimple.jar." The Voltage Web Service APIs are packaged into a viessimpl.dll package that uses the PL/SQL implementation of the SOAP client (soap_api.sql). A variety of other files and packages are included in the distribution to make implementing Voltage SecureData encryption services with Oracle PL/SQL easier.

Figure 3 VibeSimple PL/SQL for Oracle Databases

Using the Voltage User Defined Functions for Teradata

Voltage provides a set of User Defined Functions (UDF) that allow you to protect and access data stored in a Teradata database. User-defined functions (UDF) are Teradata’s mechanism to allow end-users to incorporate new functions into their database nodes and invoke them in any context that allows standard SQL functions. The formats used for protection are obtained from the Voltage SecureData server.

Figure 4 Voltage UDFs for Teradata

Voltage SecureData supports Tokenization. The token along with the encrypted data item such as a credit card numbers are stored in a database (Figure 5 and Figure 6).

---

1 Voltage SecureData Developer Guide, Version 4.0.1a, page 4-1 (ATT000358)
2 Voltage SecureData Developer Guide, Version 4.0.1a, page 5-1 (ATT000376)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Understanding Tokenization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tokenization is an alternate way to protect data and limit the scope of PCI audits. When an application or database uses tokens, instead of actual account numbers, that system generally falls outside the scope of a PCI audit. As a result, organizations that use Voltage SecureData Tokenization can reduce the size and expense of their audits.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Figure 5 Voltage SecureData Tokenization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Database Requirements for Tokenization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A database table is created in the defined tokenization database each time a tokenization format is created. The database parameters are entered when the database resource is created in the Management Console. The user name specified when the resource is created must be able to CREATE tables, INDEX, and VIEW the database, not just SELECT, INSERT, UPDATE, and DELETE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Figure 6 Tokenization Database Requirements</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 Voltage SecureData Architecture Guide, October 2011, Version 5.0, page 1-12, ATTO01041
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>storing the data as encrypted data element values (DV) in records (P) in a first database (O-DB),</td>
<td><strong>Construction:</strong> Record: &quot;the contents of the cells in a single row, optionally including any record identifier in that row.&quot; <strong>Data element value:</strong> &quot;a value which in a given record specifies a data element type.&quot; <strong>Column 2, Line 41</strong> &quot;Thus, the invention provides a method for processing of data that is to be protected, comprising the measure of storing the data as encrypted data element values of records in a first database (O-DB) each data element value.&quot;</td>
<td>Voltage SecureData stores data as encrypted data element values in records in a first database. Figure 7 shows that SecureData stores encrypted data within a database. The database stores the encrypted data element values within records. This is the first database.</td>
</tr>
</tbody>
</table>

The Voltage SecureData solution is designed to provide encryption of data, including credit card numbers, Social Security numbers and other data stored in databases and applications. The Voltage SecureData SOA Encryption Web Services are part of the Voltage SecureData server and provide a Web Services encryption interface that can be used by any application capable of consuming WSDL information. The encryption functions have also been specifically adapted for PL/SQL and Teradata applications.

**Figure 7 Storing Encrypted Data in a Database**

Voltage SecureData works with Oracle, TeraData, and other relational databases. These databases store data to be protected. A record is one row of protected data in the first database.

For Voltage Tokenization, SecureData creates a table in the database with the structure shown in Figure 8. The input data, such as a credit card number, is encrypted with an AES encryption key and random Initialization Vector (IV) and stored in the "vs_key" column. The token, for this case the credit card token, is stored in the "vs_val" column. An example

---

7 Voltage SecureData Developer Guide, Version 4.0.1a, page 1-1 (ATT000262)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>being, linked to a corresponding data element type.”</td>
<td></td>
<td>table is shown in VOLTAGE003175.</td>
</tr>
</tbody>
</table>

```sql
CREATE TABLE VOLTAGE.VS_LOOKUP_99
(
    VS_KEY VARCHAR(2048 BYTE),
    VS_KEY_IV VARCHAR(64 BYTE),
    VS_KEY_MAC VARCHAR(128 BYTE),
    VS_VAL VARCHAR(2048 BYTE),
    VS_VAL_IV VARCHAR(64 BYTE),
    VS_VAL_MAC VARCHAR(128 BYTE),
    VS_VERSION INTEGER,
    VS_CREATION_TIME TIMESTAMP(6),
    VS_UPDATE_TIME TIMESTAMP(6),
    VS_TWEAK VARCHAR(1024 BYTE),
    VS_SPARE VARCHAR(1024 BYTE)
);
```

- `vs_key_iv` = Random IV value, used to randomize the AES encryption of the input data
- `vs_key` = Input data encrypted with the AES encryption key and the random IV.
- `vs_key_mac` = HMAC (one-way hash) of input data
- `vs_val_iv` = Random IV value, used to randomize the AES encryption of the token
- `vs_val` = Randomly generated token encrypted with the AES encryption key and the random IV.
- `vs_val_mac` = HMAC (one-way hash) of randomly generated token
- `vs_version` = version number of rollover keys used to encrypt/decrypt the data.
- `vs_creation_time` = timestamp when random token was generated.
- `vs_update_time` = timestamp when data for this record was re-encrypted during rollover.
- `vs_tweak` = currently not used
- `vs_spare` = currently not used

---

8 Voltage SecureData Tokenization Deployment Best Practices, Rev. 2.1, page 11

---

©2012 TAEUS International Corporation
All Rights Reserved
www.TAEUS.com
When a security administrator sets up a data type for tokenization through the SecureData Management Console, a tokenization database table is created in the selected database, Figure 9.

![Diagram of SecureData Creating a Tokenization Database](image)

When you create a format that uses the tokenization database, a tokenization database table is created immediately in the specified resource. If you try to deselect Use Tokenization Database in a format you are editing, before you have removed the tokenization table from the database, an error is displayed. Delete the tokenization table first.

---

9 Voltage SecureData Tokenization Deployment Best Practices, Revision 2.1, pages 8 & 9 (VOLTAEG003172-3)
10 Voltage SecureData Administrator Guide, Version 5.0, page 82 ATT001197
Voltage's SecureData Database System Product Elements

Voltage SecureData Products have a first database with rows and columns, in which each row of protected data represents a record and each combination of a row and a column representing a data element value. In this database, each data element value to be protected is linked to a corresponding data element type. This is evidenced as follows:

Oracle, TeraData, and other relational databases that SecureData works with store data to be protected (the Credit Card No column) in a first database composed of rows and columns. Each row of protected data in the first database represents a record, and each combination of a row and a column represents a data element value.

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Date</th>
<th>Credit Card No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill</td>
<td>Smith</td>
<td>06-01-2012</td>
<td>1234-4567-9876-4321</td>
</tr>
<tr>
<td>Bob</td>
<td>Jones</td>
<td>07-02-2011</td>
<td>2345-6789-8765-5432</td>
</tr>
</tbody>
</table>

Example Database Table

Each data element value to be protected is linked to a corresponding data element type. In the example table, the credit card number, 1234-4567-9876-4321, is linked to the “Credit Card No” type.

The table structure for the Voltage Tokenization solution is shown in Figure 8. An example table for credit cards is shown in VOLTAGE003175\(^4\). The data element values in the "vs_key" column are linked to the data type associated with the data element values, in the example given, the data element type will be: credit-card.

\(^11\) Voltage SecureData Tokenization Deployment Best Practices, Rev. 2.1, page 11
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>comprising the measure of storing the data as encrypted data element values of records in a first database (0-DB) each data element value being, linked to a corresponding data element type.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Column 3, Line 1**
"Data element type" identifies a specific category of data. For example, identification information (name and address) could be a particular data element type. Whereas, some descriptive information (social allowance) could be a different data element type, and other descriptive information could be yet another different data element type. "Data element value"
|--------------------------------------|-------------------------------------------------|-------------------------------------------------|
| concerns a value which in a given record specifies a data element type. | **Construction:** "A data element protection catalogue (DPC), which contains --- in a location other than a column heading of a table --- each individual data element type (DT) and one or more protection attributes stating processing rules for data element values (DV), which in the first database (O-DB) are linked to the individual data element type (DT);" | Voltage SecureData has a second database (IAM-DB) that stores a data element protection catalogue, which contains each individual data element type and one or more protection attributes stating processing rules for data element values. These data element values are linked to the individual data element type in the first database for each user-initiated measure aiming at processing of a given data element value in the first database. This is evidenced as follows:

The Voltage SecureData Server, Figure 2, contains a database. Analysis of the SecureData Server software shows that the SecureData Server uses a SQLite (www.sqlite.org) database managed by Django. Django is a high-level Python (a software programming language) Web framework that contains a database access API (www.djangoproject.com). The SQLite database is a second database. The "settings.py" source code file in the SecureDataAppliance-3.4dps_server-trunk-r99215\console\key_mgmt directory contains the lines (starting at line 18):

```
DATABASE_ENGINE = 'sqlite3'
DATABASE_NAG = '..\sql\sqetel ese.sql3'
```

The same reference was found in the "settings.py" source code file in the SecureDataAppliance-3.3.2\dps-server-3.3\console\key_mgmt directory.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>meaning.</td>
<td>Data element value: “a value which in a given record specifies a data element type.”</td>
<td>The SecureData Management Console, Figure 2, allows a security administrator to set protection attributes for data types to be protected. Protection attributes may be setup for the following data types (ATT001177)\textsuperscript{12}:</td>
</tr>
</tbody>
</table>
| Column 2, Line 48                    | “storing in a second database (IAM-DB) a data element protection catalogue, which for each individual data element type contains one or more protection attributes stating processing rules for data element values, which in the first database are linked to the individual data element type;” | • Credit card numbers  
• U.S. Social Security numbers  
• Variable-length strings (VLS)  
• Specified-format strings (SFS)  
• Numbers  
• Dates |

\textsuperscript{12} Voltage SecureData Administrator Guide, Version 5.0, page 62

Figure 21 shows an example set of format settings or protection attributes that may be set through the SecureData Management Console. The “Format Name” is used to link the protection attributes established in the SecureData Server and stored in the SecureData Server’s database (the second database) with the data element types of protected data in the first database.

The “Data Protection Type” establishes the type of encryption algorithm used to encrypt and decrypt the data element type. The “Output Alphabet” establishes what characters may appear in FPE (Format Preserving Encryption) encrypted data. The “Data Protection Type” and “Output Alphabet” are two of SecureData’s protection attributes that state rules for processing data. Other protection attributes include masking parameters.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Voltage’s SecureData Database System Product Elements</strong></td>
</tr>
<tr>
<td></td>
<td>The SecureData Server’s database contains a data protection catalog which stores the protection attributes for each data element type. The SecureData server software contains a Python class, <code>FormatSpec</code>, defined in the file (there is a similar file in the SecureDataAppliance 3.3.2 source code):</td>
</tr>
<tr>
<td></td>
<td><code>SecureDataAppliance-3.4\dps_server-trunk-r99215\console\soa\models.py</code></td>
</tr>
<tr>
<td></td>
<td>The Python class, <code>FormatSpec</code>, defines a database table for storing data type and data format information within the SQLite database. The <code>FormatSpec</code> table has columns as shown in the Table 1 (not all of columns found are shown). The <code>FormatSpec</code> table is part of the data protection catalog stored in the SQLite database. There are other tables defined within the SQLite database in the SecureData Key Management Server. These tables contain additional information about encryption keys, “Districts”, key rotation groups and other information related to protection attributes. These tables are part of the data protection catalog containing protection attributes that state rules for processing data that is to be protected.</td>
</tr>
<tr>
<td></td>
<td>The SecureData Server makes a portion of the data protection catalog available by publishing configuration files, <code>clientPolicy.xml</code> and <code>vsconfig.txt</code> (Figure 10 and Figure 11).</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Data Protection Settings</strong></td>
<td></td>
</tr>
</tbody>
</table>

All of the settings in this section apply to both the Voltage SecureData Web Services, the Voltage SecureData CL, Voltage SecureData for Payments, and the Voltage SecureData Simple API. By configuring these settings once here, you can be certain that all of your formats, and other system settings are consistent across applications and batch operations used to perform encryption. In addition, you only need to configure these settings once instead of having to do it separately for each interface.

The following can be performed:

- Format Settings
- Configuring Global System Settings
- Tokenization Rollover Settings

**Figure 10 SecureData Data Protection Settings**

<table>
<thead>
<tr>
<th><strong>Using Formats</strong></th>
<th></th>
</tr>
</thead>
</table>

Formats created in the Voltage SecureData Management Console are available for clients to use by parsing the `clientPolicy.xml` or `vsconfig.txt` files published at deployment time. The Web Service gets its format information through an XML file that is also published by the Console when you deploy your changes.

**Figure 11 Protection Attribute Availability**

An example `clientPolicy.xml` file named "keynumberpolicy.xml" (pages 1 – 9) was found with the Voltage source code. (Because of the protection order restrictions, a copy of the keynumberpolicy.xml file was not available to include with the report). This `clientPolicy`
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration file contains in part (the specific xml format is not include below, refer to the keynumberpolicy.xml file for specific details):</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>encryptionSettings</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o DefaultFPETime value=&quot;090101000000Z&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o DefaultKeySize value=&quot;192&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o USSSNFirstFiveOnly true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o CreditCardNumberMaskInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ showLeading=&quot;1&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ showTrailing=&quot;4&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ maskCharacter=&quot;#&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ highestAccessLevel=&quot;true&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o SocialSecurityNumberMaskInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ showLeading=&quot;1&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ showTrailing=&quot;4&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ maskCharacter=&quot;#&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ highestAccessLevel=&quot;true&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o FormatMappings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ formatName=&quot;Custom-CreditCard&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ value=&quot;0-9&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ inputType=&quot;String&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ rangeFormat=&quot;true&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ unicode=&quot;false&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ ignoreOutOfRange=&quot;true&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ MaskInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o showLeading=&quot;0&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>o showTrailing=&quot;5&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>for each user-initiated measure aiming at processing of a given data element value</td>
<td>Construction: Plain meaning of “initially producing a calling to the data element protection”</td>
<td>Voltage SecureData initially produces a calling to the data element protection catalogue for collecting the protection attribute/attributes associated with the corresponding data element type. This is evidenced as follows:</td>
</tr>
</tbody>
</table>

- keyNumberConfig
  - keyNumberTable
    - keyNumber number="1"
    - commonName="mask@dataprotection.voltage.com"
    - time="080101000000Z"
    - district="dataprotection.voltage.com#118815279"
    - application="data"
    - algorithm="FPE"
    - keySize="256"
    - policy=""
    - lable=""
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(DV) in the first database (O-DB), initially producing a calling to the data element protection catalogue for collecting the protection attribute/attributes associated with the corresponding data element type, and catalogue for collecting the protection attribute/attributes associated with the corresponding data element type.&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Plain meaning of the word “user” which I do not take to be limited to persons. Column 2, Line 54 “in each user-initiated measure aiming at processing of a given data element value in the first database, initially producing a compelling calling to the data element protection catalogue for collecting the protection attribute/attributes associated with the corresponding data element type, and compellingly |

To process (encrypt, decrypt, or decrypt with masking) a data element value, an encryption key must be obtained from the Voltage SecureData Server. Voltage software modules do provide means for caching encryption keys, but the encryption key must still be obtained from the Voltage SecureData Server at least the first time. Also, encryption and decryption process may occur locally at the client application by obtaining an encryption key from the Voltage SecureData Server or the data to be encrypted or decrypted may be sent to a service module on the SecureData Server in which case the encryption/decryption process will occur at the SecureData Server. In either case, the process of obtaining an encryption key and the encryption/decryption process are essentially the same.

Voltage SecureData uses an encoded identity to obtain an encryption key from the Voltage SecureData Server. A client application starts with a key name or identity that has the form of an email address. An encoded identity is generated by software modules provided by Voltage. The software modules take the key name or identity along with the data type and a time stamp and then extract protection attributes from the clientPolicy.xml and/or vsconfig.txt configuration file(s) obtained from the SecureData Server in order to generate the encoded identity (Figure 12 and Figure 13). Extracting protection attributes from the configuration information specified through the Management Console (the DPC) is initially producing a calling to the data protection catalog (DPC) to collect protection attributes.
controlling the processing of the given data element value in conformity with the collected protection attribute/attributes."

---


©2012 TAEUS International Corporation
All Rights Reserved
www.TAEUS.com
### Table 2-1 Identity Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name</td>
<td>The common name field is the foundation of the identity. Currently, the only type of common name supported by Voltage SecureData is in the form of an email address, which is case-insensitive.</td>
</tr>
<tr>
<td>Identitytimes</td>
<td>When the time field is used with symmetric protection, the date is a notBefore time and acts as another field that defines the key; changing the time is a way to roll over keys.</td>
</tr>
<tr>
<td>District</td>
<td>The domain name plus serial number. For more information about districts, see &quot;Understanding Districts&quot; on page 2-7.</td>
</tr>
<tr>
<td>Application</td>
<td>Specifies what type of data Voltage SecureData is protecting: disk data, database fields, etc. If not specified, the default is email.</td>
</tr>
<tr>
<td>Algorithm</td>
<td>An identifier of the algorithm used to protect the data. In this way, if the rest of the identity information (email address, key size, etc.) is the same, the key is still different for different protection algorithms.</td>
</tr>
<tr>
<td>Key Size</td>
<td>If the algorithm identifier specifies a key size, the key size field could be Triple-DES, AES-128-CBC, etc.</td>
</tr>
</tbody>
</table>

**Figure 14 Encoded Identity Fields**

Voltage supplies a software module: VibeSimple PL/SQL Package. This package is designed to work with Oracle databases, and is designed to make it easy to implement encryption services such as transparent encryption and decryption of data element values within an Oracle database (Figure 15). The VibeSimple PL/SQL Package provides functions that can be added to database view and triggers. The functions added to views and triggers cause user initiated measures aimed at processing (encrypting or decrypting) data element values to use Voltage Web Service APIs to encrypt or decrypt the data element values.
Using the VibeSimple PL/SQL Package

Voltage provides an implementation of VibeSimple for PL/SQL. The VibeSimple functions are described in Chapter 3, “VibeSimple jar.” The Voltage Web Service APIs are packaged into a vibesimple PL/SQL package that uses the PL/SQL implementation of the SOAP client (soap_api.sql). A variety of other files and packages are included in the distribution to make implementing Voltage SecureData encryption services with Oracle PL/SQL easier.

Figure 15 VibeSimple PL/SQL Package for Oracle Databases

The VibeSimple PL/SQL Package include functions for encrypting and decrypting specific data types (Voltage SecureData Developer Guide) such as:

- `accessCreditCardNumber(...)`: which makes a calling to decrypt a credit card number.
- `protectCreditCardNumber(...)`: which makes a calling to encrypt a credit card number.
- `accessSocialSecurityNumber(...)`: which makes a calling to decrypt a social security number.
- `protectSocialSecurityNumber(...)`: which makes a calling to encrypt a social security number.

The VibeSimple PL/SQL Package come with functions for encrypting and decrypting data types that are setup through the SecureData Management Console.

---

1 Voltage SecureData Developer Guide, Version 4.0.1a, page 4-1, ATT000358
|---------------------------------------|--------------------------------------------------------|--------------------------------------------------|
|                                       |                                                        | • `accessFormattedData(…)` which makes a calling to decrypt a data element type. The data element type is passed into the `accessFormattedData(…)` function via the “format” parameter.  
• `protectFormattedData (…)` which makes a calling to encrypt a data element type. The data element type is passed into the `protectFormattedData (…)` function via the “format” parameter. |

Figure 16 shows a database view that supports the automatic encryption of a data element value. The steps in the process are:

1. A user (the application) initiates a measure to store or modify a payment data element value via an SQL request.
2. A trigger captures the request and makes a call to the Voltage SecureData SOA via a PL/SQL function provided by Voltage.
3. The SecureData SOA makes a calling to the `clientPolicy.xml` configuration file (a portion of the data protection catalog) to collect protection attributes associated with the data element type. After the SecureData SOA obtains the protection attributes, the data to be protected is encrypted in conformity with the collected protection attributes.
4. Finally, the encrypted value is stored as an encrypted data element value in the first database.

---

19 Voltage SecureData Developer Guide, Version 4.0.1a, page 4-10 & 11, ATT000367-8
### Voltage’s SecureData Database System Product Elements

|----------------------------------------|------------------------------------------------------------|-----------------------------------------------------|

1. **SQL Request sets off Trigger**

   - **Database Server**
   - **PaymCrypt**
   - **Payments**
   - **Trigger**
   - **Application**

2. **Trigger calls SOA to encrypt**
   - **SOA sends back encr. data**
   - **Voltage SecureData SOA**

3. **Store encr. data**

4. **Unencrypted data**

5. **Encrypted data**

---

**Figure 16 A Database Trigger for Transparent Encryption of Data**

Figure 17 shows a database view that supports the automatic decryption of a data element value. The Voltage SOA makes a calling to the clientPolicy.xml configuration file (a portion of the data protection catalog) to collect protection attributes associated with the data element type. After the SecureData SOA obtains the protection attributes, the data to be protected is decrypted in conformity with the collected protection attributes.
### Voltage’s SecureData Database System Product Elements

<table>
<thead>
<tr>
<th>Database Server</th>
<th>PaymCrypt</th>
<th>Payments</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Encrypted data is retrieved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. View calls the SOA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Web service returns plaintext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Plaintext sent to application</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram of database view](image)

**Figure 17 A Database View for Transparent Decryption of Data**

Voltage supplies a set of User Defined Functions (UDFs) for Teradata Databases. This set of UDFs is designed to make it easy to implement encryption services such as transparent encryption and decryption of data element values within Teradata database (Figure 18). The UDFs provides functions that can be added to database views and triggers. The functions added to views and triggers cause user initiated measures aimed at processing (encrypting or...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>decrypting) data element values to use Voltage SecureData services to encrypt or decrypt the data element values.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Using the Voltage User Defined Functions for Teradata

Voltage provides a set of User Defined Functions (UDF) that allow you to protect and access data stored in a Teradata database. User-defined functions (UDF) are Teradata’s mechanism to allow end-users to incorporate new functions into their database nodes and invoke them in any context that allows standard SQL functions. The formats used for protection are obtained from the Voltage SecureData server.

**Figure 18 Voltage User Defined Functions for TeraData**

There are two UDFs supplied by Voltage that may be used with views and triggers for encrypting and decrypting data element values:

- vsAccessFormattedData(...) which makes a calling to decrypt a data element type. The data element type is passed into the accessFormattedData(...) function via the “formatIn” parameter.
- vsProtectFormattedData (...) which makes a calling to encrypt a data element type. The data element type is passed into the vsProtectFormattedData (...) function via the “formatIn” parameter.

---

22 Voltage SecureData Developer Guide, Version 4.0.1a, page 5-1, ATT000376
23 Voltage SecureData Developer Guide, Version 4.0.1a, page 5-12 & 13, ATT000387-8
|--------------------------------------|---------------------------------------------------------------|-----------------------------------------------------|

The use of the UDFs within database views and triggers is the same as that outlined in Figure 17 and Figure 16.

In Voltage SecureData, tokenization uses the same processes outlined above for encrypting and decrypting a data item to be protected, such as a credit card numbers. The Voltage SecureData Web Services are used for the encryption and decryption processes. The ProtectFormattedData and AccessFormattedData functions are used to encrypt and decrypt the protected data element values. The “format” parameter is the date element type used to link to the data element type of the data element value. The functions must make a calling to the clientPolicy.xml configuration file (a portion of the data protection catalog) to collect protection attributes associated with the data element. For tokenization when a new data value is encrypted, the ProtectFormattedData function will both encrypt the data value and generate a token for the data value. Both the encrypted data value and the token value are stored in the database.

---

24 Voltage SecureData Developer Guide, Version 4.0.1a, page 2-17 - 21, ATT000290 - 4
25 Voltage SecureData Tokenization Deployment Best Practices, Rev. 2.1, page 11
### US Patent 6,321,201 Claim 1 Limitations

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tokenization Formats</strong>&lt;br&gt;To use the tokenization feature, you must create a format that specifies tokenization and identify the database that holds the tokens and protected data. Then you perform the protect operation using Web Services or the Voltage SecureData Command Line. The tokens are generated and stored during the protection process. Changes to the format settings do not affect how data is being tokenized/de-tokenized if tokens were already created for that data.</td>
<td><strong>Figure 19 SecureData Tokenization</strong>&lt;sup&gt;26&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

controlling the user's processing of the given data element value in conformity with the collected protection attribute/attributes.<br><br>**Construction:**<br>Plain meaning of the word “user” which I do not take to be limited to persons.<br><br>**Data element value:** “a value which in a given record specifies a data element type.”<br><br>**Column 2, Line 54**<br>“in each user-initiated measure aiming at

---

<sup>26</sup> Voltage SecureData Administrator Guide, Version 5.0, page 65 ATT001180
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>processing of a given data element value in the first database, initially producing a compelling calling to the data element protection catalogue for collecting the protection attribute/attributes associated with the corresponding data element type, and compellingly controlling the processing of the given data element value in conformity with the collected protection attribute/attributes.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2012 TAEUS International Corporation
All Rights Reserved
www.TAEUS.com
UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF CONNECTICUT

--------------------------------------------------X

PROTEGRITY CORPORATION

Plaintiffs.

v.

Case No.: 3:10-CV-00755

VOLTAGE SECURITY, INC.

Defendants.

--------------------------------------------------X

EXPERT REPORT OF HARRY DIREEN, PH.D., P.E.

THIS REPORT CONTAINS CONFIDENTIAL AND HIGHLY CONFIDENTIAL
INFORMATION SUBJECT TO PROTECTIVE ORDER DATED 19 JANUARY, 2012
I. INTRODUCTION ............................................................................................................................................. 3

II. BACKGROUND AND QUALIFICATIONS ..................................................................................................... 3

III. PROJECT DESCRIPTION ............................................................................................................................. 5

IV. APPROACH .................................................................................................................................................. 6

V. INFORMATION CONSIDERED AND MATERIALS REVIEWED ................................................................. 7
   A. Patents ....................................................................................................................................................... 7
   B. Documents ............................................................................................................................................... 7
   C. Court Documents ...................................................................................................................................... 7
   D. Other Documents and Books .................................................................................................................... 7
   E. Source Code Inspections ........................................................................................................................... 8

VI. OPINIONS REGARDING INFRINGEMENT ................................................................................................... 8
   A. Overview of the Protegrity Patents ............................................................................................................. 8
   B. Voltage Security, Inc. SecureData ............................................................................................................. 10
   C. Infringement Opinions ............................................................................................................................ 23

VII. DECLARATION ........................................................................................................................................... 25

Highly Confidential Information: Subject to Protective Order
DRAFT

I. INTRODUCTION

1. I, Harry G. Direen Jr., Ph.D., P.E., declare that the opinions stated in this expert report are my own, that I have personal knowledge of the facts stated in this expert report, and am competent to testify to the facts and opinions within. If called as a witness, I can and will testify to the following.

2. My full name is Harry G. Direen Jr. Counsel for Protegrity Corporation has retained me as an expert witness with respect to this matter.

3. This declaration reflects my opinions with respect to the issues that I have been asked to address. I reserve the right to supplement this expert report, if and when appropriate, based upon information I learn after the date of this declaration.

II. BACKGROUND AND QUALIFICATIONS

4. I have over 30 years of experience as an engineer designing and developing products in the electronics and software industry. I received my BS in Electrical Engineering from the University of California at Irvine and my Ph.D. in Electrical Engineering with a focus in Control Systems from the University of Colorado at Boulder. I am a registered profession engineer in the state of Colorado and a member of the IEEE. My expertise includes control systems; embedded systems design and development; XML database design; control of UAVs (Unmanned Aerial Vehicles); bio-informatics; and product reverse engineering. I have a broad base of knowledge in electronic and software systems gained in my career working on various product designs for a number of companies. I developed the embedded control of high power RF amplifiers used in Highly Confidential Information Subject to Protective Order
Magnetic Resonant Imaging systems. I was a chief architect developing a novel XML database system. I developed DNA and protein sequence search algorithms and software used for bioinformatics. I am working with a research group at the US Air Force Academy on the control of UAVs. I have reversed engineered a number of complex systems, including: media players, modems, modem compression algorithms, coriolis mass flow meters, blood analysis meters and a variety of other products. I have successfully discovered and analyzed complex digital signal processing algorithms and pertinent operations within these products. I have taught engineering at the US Air Force Academy, and taught under graduate and graduate level classes at the University of Colorado, Colorado Springs (UCCS). I was a member of Advisory Board for the Center of Computational Biology, at the University of Colorado, Denver and was a Member of UCCS Advisory Board for developing undergraduate courses in the emerging area of computational biology. I am an inventor on two patents associated with the XML database technology, and an inventor on two patents in process for products I am working on.

5. I have not testified as an expert at trial within the past four years.

6. I have testified as an expert by deposition:


7. I have read and am fully familiar with United States Patent 6,321,201 assigned to Protegrity Corporation (the Protegrity Patents). By virtue of my education, experience and training, as described above, I have expertise in the arts addressed by these patents. A complete

Highly Confidential Information: Subject to Protective Order

4
copy of my curriculum vitae is attached as Exhibit A. The compensation paid to TAEUS for this work is detailed in Exhibit D.

III. PROJECT DESCRIPTION

8. Counsel for Protegrity Corporation asked me to review Voltage Security, Inc. materials and to examine, and document the behavior of Voltage Security, Inc. products to determine certain aspects of their characteristics and features. I was also asked to compare the system architecture, operational characteristics and features of these products to selected claims of United States Patent 6,321,201 assigned to Protegrity Corporation. The Voltage Security, Inc. SecureData solution includes the products and software I inspected which are shown in Table 1. The patent claims to which I compared these products are shown in Table 2.

<table>
<thead>
<tr>
<th>Voltage SecureData Solution</th>
<th>Product / Software Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecureData Key Appliance 3.3.2</td>
<td>appliance-1.2-r57958</td>
</tr>
<tr>
<td></td>
<td>dps_server-3.3-r99208</td>
</tr>
<tr>
<td></td>
<td>VibeWebService-2.1-r56640</td>
</tr>
<tr>
<td>SecureData Key Appliance 3.4</td>
<td>appliance- trunk-r59422</td>
</tr>
<tr>
<td></td>
<td>dps_server- trunk-r99215</td>
</tr>
<tr>
<td></td>
<td>VibeWebService- trunk-r61873</td>
</tr>
<tr>
<td>SecureData SOA Web Services Server</td>
<td>SD-CL</td>
</tr>
<tr>
<td></td>
<td>SecureData-SOA-3.1</td>
</tr>
<tr>
<td></td>
<td>SecureData-SOA-3.2</td>
</tr>
<tr>
<td>SecureData Simple API 3.7-r64724</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.8-r66619</td>
</tr>
<tr>
<td>SecureData for Teradata TeradataUDF 1.1-r99219</td>
<td></td>
</tr>
<tr>
<td>Host-SDK host-sdk-2.0-r66898</td>
<td></td>
</tr>
<tr>
<td></td>
<td>host-sdk-2.2.1-r74467</td>
</tr>
<tr>
<td>Z-FPE</td>
<td>1.3.0_r61631-r99541</td>
</tr>
</tbody>
</table>

Highly Confidential Information: Subject to Protective Order
IV. APPROACH

9. First, I reviewed the complaint and the patent. I then reviewed documents produced by Voltage Security, Inc. and identified those documents that were deemed relevant to determining whether or not the accused Voltage Security, Inc. products infringe the patent. I also inspected source code identified by Voltage Security, Inc. at E-Hounds in Clearwater, FL. I also reviewed certain Voltage Security, Inc. documents in the public record, and transcripts of the deposition testimony of certain Voltage Security, Inc. witnesses. After reviewing all of this information, I compiled my findings with respect to infringement of the asserted claims of the patents and then documented my findings in the claim chart attached to this report as Exhibit B.
V. INFORMATION CONSIDERED AND MATERIALS REVIEWED

A. Patents


B. Documents

1. Exhibit C lists documents provided by counsel.
2. Datasheet: Voltage SecureData, Encrypting Content Wherever It Goes, DS020608
5. Datasheet: Voltage SecureData, End-to-End Data Protection For the Way Your Business Works,

C. Court Documents


D. Other Documents and Books

2. Stored Procedures, Triggers, and User-Defined Functions on DB2 Universal Database for iSeries, by Bedoya, Cruz, Lema & Sinkorapoom, October 2006, ibm.com/redbooks

Highly Confidential Information: Subject to Protective Order
   http://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm

E. Source Code Inspections
1. November 12, 13, & 14, 2012, Clearwater, FL.
3. December 3, 4, 2012 Clearwater, FL.
4. Source Code printouts: (waiting for Bates numbers)

VI. OPINIONS REGARDING INFRINGEMENT
A. Overview of the Protegrity Patents
10. The asserted patent in this suit is U.S. Patent Nos. 6,321,201, ("the ’201 patent" or "the patents-in-suit").

11. The ’201 patent relates to protecting sensitive data, such as credit card numbers, social security numbers, and many other types of sensitive data, within a database by encrypting the sensitive data\(^1\). The data encryption and decryption processes are handled in a way that is “transparent” to an authorized user of the data\(^2\). "Transparent" means that the authorized user, or application software, is un-aware that the encryption and decryption process happens behind the scenes and the user takes no special or additional steps to cause the encryption and decryption of the sensitive data. Using the ’201 patent technology, only sensitive data needs to be encrypted,

---
\(^1\) ’201 patent, col. 2, lines 41-44.
\(^2\) Id. at col. 4, lines 11 – 17.

Highly Confidential Information: Subject to Protective Order
non-sensitive data within the same database record can be left un-encrypted. In addition, different encryption algorithms and protection attributes can be applied to different data types within a database record. A security administrator may setup and apply protection attributes to associate with sensitive data stored within a database.

12. To effect the protection of sensitive data, the '201 patent defines a data protection catalogue (DPC). The DPC stores protection attributes which provide information on the rule or rules applied to processing the corresponding data element. The rule or rules typically involve rules for encrypting and decrypting the data element. The protection attributes may include which encryption algorithm to use along with appropriate parameters associated with the encryption process.

13. User initiated measures, such as reading or modifying a sensitive data element, will automatically cause the data protection catalog to be accessed. The protection attributes for the corresponding data type will be read from the DPC. These protection attributes will be used to determine the specific rules to apply to the data element. Typically the rules will determine how to decrypt or encrypt the data element value. Automatically applying processing rules such as encryption and decryption of sensitive data element values, without specific user interaction to affect the processing rules, is what enables the transparent processing of the sensitive data. This is the core of the invention of the '201 patent.

---

3 Id. at col. 4, lines 33-34.
4 Id. at col. 4, lines 26 – 32, and col. 11, lines 1 – 10.
5 Id. at col. 3, lines 49 – 51.
6 Id. at col. 4, lines 16 – 25.
B. Voltage Security, Inc. SecureData

14. The Voltage Security, Inc. SecureData Solution is designed to protect data stored in databases, Figure 1. The Voltage SecureData Solution is made up of a number of components as seen in Figure 2. The SecureData components of primary concern relative to the '201 patent are outlined below.

**Overview of Voltage SecureData**

The Voltage SecureData solution is designed to provide protection of data, specifically credit card numbers, Social Security numbers and other data stored in databases and applications as shown in Figure 1-1.

![Voltage SecureData Solution](image)

**Figure 1-1 Voltage SecureData Solution**

**Figure 1 Voltage SecureData Solution**

---


Highly Confidential Information: Subject to Protective Order
Figure 1-2 Voltage SecureData Solution Components

Figure 2 Voltage SecureData Solution Components

- Voltage SecureData Key Management Server: The Key Management Server is a centralized key management system that manages the generation and issuance of FPE (Format Preserving Encryption), AES (Advanced Encryption Standard), and IBE (Identity Based Encryption) encryption keys. The Key Management Server also provides user authentication.

---


Highly Confidential Information: Subject to Protective Order
- **Voltage SecureData Management Console**: The Management Console provides centralized configuration of keys, data types, and reporting across the SecureData solution. The Voltage SecureData Administrator Guide provides the full range of configuration that can be handled through the Management Console\(^9\).

- **Voltage SecureData Web Services Server**: The Web Services Server provides a Web Service data protection interface that can be used by any application capable of consuming WSDL (Web Services Description Language) information\(^10\). The protection functions supplied by SecureData have been adapted for PL/SQL applications. PL/SQL (Procedural Language / Structured Query Language) is Oracle Corporation’s extension language designed to work with the Oracle relational databases.

- **Voltage SecureData Simple API (Application Programmer’s Interface)\(^11\)**: this API provides a simple interface on top of the Voltage Encryption Toolkit. The API makes it easy for a developer to incorporate protection / access (or encryption / decryption) functionality directly into existing applications.

- **Voltage Encryption Toolkit\(^11\)**: High-performance low-level IBE Elliptic Curve Cryptography toolkit with work with standard cryptographic libraries such as

---

\(^9\) Voltage SecureData Administrator Guide, Version 5.0  

Highly Confidential Information: Subject to Protective Order
AES, SHA256, RSA, etc. The Toolkit interfaces directly with the Voltage
Voltage SecureData Key Management Server.

• Voltage POS SDK\textsuperscript{11}: A light-weight C SDK with a subset of low-level IBE
  Elliptic Curve Cryptography toolkit protection functions specifically designed to
  be embedded directly in Point of Sales (POS) devices or software-based ECR
  systems.

• Voltage Host SDK\textsuperscript{11}: A subset of low-level IBE Elliptic Curve Cryptography
  toolkit data protection functions specifically designed to be incorporated into a
  payments host. These APIs take data and keys protected with the POS API and
  access and then reprotect the data before it enters the payment providers database.

• Voltage SecureData VibeSimple PL/SQL Package: SecureData provides a Web
  Service API (Application Programmer's Interface) which is designed to work
  with Oracle Databases\textsuperscript{12}. The PL/SQL Package provide functions for encrypting
  and decrypting data within an Oracle database by making Web Service calls to the
  Voltage SecureData Web Services Server.

\textsuperscript{12} Voltage SecureData Developer Guide, January 2001, Version 4.0.1a, page
• Voltage SecureData for Teradata: SecureData provides a set of User Defined Functions (UDFs) that encrypt and decrypt data within a Teradata database\textsuperscript{13}. These UDFs use services provide by the SecureData Key Management Server.

• Voltage SecureData Tokenization: SecureData provides a complete Tokenization solution through their Web Services interface (refer to “Understanding Tokenization, in the SecureData Architecture Guide\textsuperscript{14}). The SecureData Tokenization solution works with many databases that have a JDBC (Java Database Connectivity) interface.

15. SecureData provides means for a security administrator to setup data types to be protected through the SecureData Management Console. Section Data Protection Settings in the SecureData Administrator Guide\textsuperscript{15} covers setting up data types. The different data types that can be setup via the Management Console are:

• Credit card numbers
• U.S. Social Security numbers
• Variable-length strings
• Specified-format strings
• Numbers
• Dates

\textsuperscript{13} Voltage SecureData for Teradata Developer Guide, December 2011, Version 1.1.2, page 1-1
\textsuperscript{14} Voltage SecureData Architecture Guide, October 2011, Version 5.0, page 1-12
\textsuperscript{15} Voltage SecureData Administrator Guide, Version 5.0, page 61

Highly Confidential Information: Subject to Protective Order
Figure 3 Variable Length String Format a Management Console for setting up a variable-length string. A data type includes a “Format Name” which is used by applications to reference, or link to the specific format. A variety of parameters associated with the user defined type are set through the console, including the “Data Protection Type” which is the type of encryption or tokenization that will be used with the data type. The encryption types are FPE or eFPE. The eFPE encryption type embeds a key number into the encrypted data. If Tokenization is selected, SecureData will setup a database table in the defined database to store tokenization information. Tokenization is only available for data types (or formats) that are setup through the Management Console. Once a data type format is specified, the security administrator can establish a variety of mask features associated with the data type, as shown in Figure 3.

16 Voltage SecureData Administrator Guide, Version 5.0, page 70

Highly Confidential Information: Subject to Protective Order
Figure 3 Variable Length String Format

16. The SecureData Key Management Server contains a database for storing data type information along with a variety of other information. Analysis of the SecureData Server

\footnote{Voltage SecureData Administrator Guide, Version 5.0, page 66}

Highly Confidential Information: Subject to Protective Order
software shows that SecureData uses a SQLite database (www.sqlite.org) managed by Django. Django is a high-level Python (a software programming language) Web framework that contains a database access API (www.djangoproject.com). The SecureData server software contains a Python class, `FormatSpec`, which defines a table for storing data type and data format information within the SQLite database. The `FormatSpec` table has columns as shown in the table below (not all of the columns found are shown). The `FormatSpec` table is part of the data protection catalog stored in the SQLite database. There are other tables defined within the SQLite database in the SecureData Key Management Server. These tables contain additional information about encryption keys, districts (a construct used by SecureData for managing keys), key rotation groups and other information related to protection attributes. These tables are part of the data protection catalog containing protection attributes that state rules for processing data that is to be protected.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Data type name, used to reference this data type</td>
</tr>
<tr>
<td>formatType</td>
<td>VAR String, FIX String, Credit Card number, US SSN</td>
</tr>
<tr>
<td>inputType</td>
<td>String, Oracle Date, Number</td>
</tr>
<tr>
<td>unicode</td>
<td>Boolean to determine if string type is unicode</td>
</tr>
<tr>
<td>ignoreOutOfRange</td>
<td></td>
</tr>
<tr>
<td>charset</td>
<td>Input alphabet</td>
</tr>
<tr>
<td>outputCharset</td>
<td></td>
</tr>
<tr>
<td>maxKeyNumCount</td>
<td>Used for maximum key rotation number in eFPE</td>
</tr>
<tr>
<td>formatString</td>
<td></td>
</tr>
<tr>
<td>minValue</td>
<td>Minimum value for a number</td>
</tr>
<tr>
<td>maxValue</td>
<td>Maximum value for a number</td>
</tr>
<tr>
<td>retain_leading</td>
<td>For credit cards the number of leading numbers to retain</td>
</tr>
<tr>
<td>retain_trailing</td>
<td>For credit cards the number of trailing numbers to retain</td>
</tr>
<tr>
<td>lookup_db</td>
<td>Database for Tokenization</td>
</tr>
<tr>
<td>lookup_table</td>
<td>Database Table for Tokenization</td>
</tr>
<tr>
<td>lookup_identity</td>
<td>Identity used for Tokenization</td>
</tr>
</tbody>
</table>

Highly Confidential Information: Subject to Protective Order
17. The SecureData Key Management Server publishes portions of the data protection catalog in configuration files clientPolicy.xml and vsconfig.txt, as shown in Figure 4 and Figure 5.

![Data Protection Settings](image)

**Data Protection Settings**

All of the settings in this section apply to both the Voltage SecureData Web Services, the Voltage SecureData CL, Voltage SecureData for Payments, and the Voltage SecureData Simple API. By configuring these settings once here, you can be certain that all of your formats, and other system settings are consistent across applications and batch operations used to perform encryption. In addition, you only need to configure these settings once instead of having to do it separately for each interface.

The following can be performed:
- Format Settings
- Configuring Global System Settings
- Tokenization Rollover Settings

**Figure 4 SecureData Data Protection Settings\(^{18}\)**

![Using Formats](image)

**Using Formats**

Formats created in the Voltage SecureData Management Console are available for clients to use by parsing the clientPolicy.xml or vsconfig.txt files published at deployment time. The Web Service gets its format information through an XML file that is also published by the Console when you deploy your changes.

**Figure 5 Protection Attribute Availability\(^{19}\)**

18. The Voltage SecureData system provides UDFs (User Defined Functions) for Teradata databases (Figure 6) and PL/SQL functions (Figure 7) for Oracle databases that enable transparent encryption and decryption of data stored in databases. Voltage SecureData describes

---

\(^{18}\) Voltage SecureData Administrator Guide, Version 5.0, page 61 ATT001176

\(^{19}\) Voltage SecureData Administrator Guide, Version 5.0, page 64
how to use UDFs and PL/SQL functions for transparent encryption and decryption of data to be protected in their document *Database Integration for Voltage SecureData SOA*. The steps in this process are outlined below.

**Using the Voltage User Defined Functions for Teradata**

Voltage provides a set of User Defined Functions (UDF) that allow you to protect and access data stored in a Teradata database. User-defined functions (UDF) are Teradata’s mechanism to allow end-users to incorporate new functions into their database nodes and invoke them in any context that allows standard SQL functions. The formats used for protection are obtained from the Voltage SecureData server.

*Figure 6 Voltage UDFs for Teradata*\(^\text{20}\)

**Using the VibeSimple PL/SQL Package**

Voltage provides an implementation of VibeSimple for PL/SQL. The VibeSimple functions are described in Chapter 3, "VibeSimple.jar." The Voltage Web Service APIs are packaged into a vibesimple PL/SQL package that uses the PL/SQL implementation of the SOAP client (soap_api.sql). A variety of other files and packages are included in the distribution to make implementing Voltage SecureData encryption services with Oracle PL/SQL easier.

*Figure 7 VibeSimple PL/SQL for Oracle Databases*\(^\text{21}\)

Figure 8 shows a database view that supports the automatic encryption of a data element value. The steps in the process are:

1. A user (the application) initiates a measure to store or modify a payment data element value via an SQL request.

\(^{20}\) Voltage SecureData Developer Guide, Version 4.0.1a, page 5-1

\(^{21}\) Voltage SecureData Developer Guide, Version 4.0.1a, page 4-1

Highly Confidential Information: Subject to Protective Order
2. A trigger captures the request and makes a call to the Voltage SecureData SOA via a PL/SQL function provided by Voltage.

3. The SecureData SOA makes a calling to the clientPolicy.xml configuration file (a portion of the data protection catalog) to collect protection attributes associated with the data element type. After the SecureData SOA obtains the protection attributes, the data to be protected is encrypted in conformity with the collected protection attributes.

4. Finally, the encrypted value is stored as an encrypted data element value in the first database.

![Database Trigger for Transparent Encryption of Data](image)

**Figure 8 A Database Trigger for Transparent Encryption of Data**

Figure 9 shows a database view that supports the automatic decryption of a data element value. The Voltage SOA makes a calling to the clientPolicy.xml configuration file (a portion of the data protection catalog) to collect protection attributes associated with the data element type. After the SecureData SOA obtains the protection attributes, the data to be protected is decrypted in conformity with the collected protection attributes.

Highly Confidential Information: Subject to Protective Order

20
Figure 9 A Database View for Transparent Decryption of Data\(^{22}\)

19. **SecureData Tokenization:** When a security administrator sets up a data type to be tokenized, SecureData creates a table in the database with the structure shown in Figure 10. The input data, such as a credit card number, is encrypted with an AES encryption key and random Initialization Vector (IV) and stored in the "vs_key" column. The token, which for this case is the credit card token that is a randomly generated value, is stored in the "vs_val" column. An example table is shown in Voltage SecureData Tokenization Deployment Best Practices, Revision 2.1, page 11 (VOLTAGEN 003175).

\(^{22}\)Database Integration for Voltage SecureData SOA, page 4

Highly Confidential Information: Subject to Protective Order
CREATE TABLE VOLTAGE.VS_LOOKUP_99
{
    VS_KEY VARCHAR2(2048 BYTE)
    VS_KEY_IV VARCHAR2(64 BYTE)
    VS_KEY_MAC VARCHAR2(128 BYTE)
    VS_VAL VARCHAR2(2048 BYTE)
    VS_VAL_IV VARCHAR2(64 BYTE)
    VS_VAL_MAC VARCHAR2(128 BYTE)
    VS_VERSION INTEGER
    VS_CREATION_TIME TIMESTAMP(6)
    VS_UPDATE_TIME TIMESTAMP(6)
    VS_TWEAK VARCHAR2(1024 BYTE)
    VS_SPARE VARCHAR2(1024 BYTE)
};

- vs_key_IV = Random IV value, used to randomize the AES encryption of the input data
- vs_key = input data encrypted with the AES encryption key and the random IV.
- vs_key_mac = HMAC (one-way hash) of input data
- vs_val_IV = Random IV value, used to randomize the AES encryption of the token
- vs_val = Randomly generated token encrypted with the AES encryption key and the random IV.
- vs_val_mac = HMAC (one-way hash) of randomly generated token
- vs_version = version number of rollover keys used to encrypt/hash the data.
- vs_creation_time = timestamp when random token was generated.
- vs_update_time = timestamp when data for this record was re-encrypted, during rollover.
- vs_tweak = currently not used
- vs_spare = currently not used

Figure 10 Tokenization Database Table Structure

20. The DataSecure ProtectFormattedData web service call is used to encrypt a data value which is to be protected, such as a credit card number. The ProtectFormattedData makes a call to the SecureData data protection catalog, based upon the data type (format), to obtain protection attributes for the data value. The data value will be encrypted and stored in the vs_key column in the first database. A randomly generated

---

23 Voltage SecureData Tokenization Deployment Best Practices, Rev. 2.1, pages 8 & 9
24 Voltage SecureData Developer Guide, Version 5.0, page 2-17

Highly Confidential Information: Subject to Protective Order

22
token will also be created for the data value and this token value will be stored in the vs_val column in the first database. The ProtectFormattedData function will then return the token value.

21. The DataSecure AccessFormattedData\textsuperscript{25} web service call is used to decrypt a protected value. The AccessFormattedData function makes a call to the SecureData data protection catalog based on the data type (format) to obtain protection attributes for the data value. The encrypted data value corresponding to the token is retrieved from the first database, and then the encrypted data value is decrypted based on the protection attributes. The decrypted value is then returned to the user.

C. Infringement Opinions

22. The claim charts attached as Exhibit B contain my findings with respect to the literal infringement of the Protegrity '201 Patent by the Voltage Security, Inc. products that are identified in these charts of the asserted claims of the patents-in-suit as the Court has construed these claims in its Markman Order. I reserve the right to supplement these findings with respect to the infringement by these Voltage Security, Inc. products of the asserted claims of the patents-in-suit under the doctrine of equivalents, to the extent appropriate in light of any challenge Voltage Security, Inc. may make to my findings of literal infringement.

\textsuperscript{25} Voltage SecureData Developer Guide, Version 5.0, page 2-20

Highly Confidential Information: Subject to Protective Order
23. As set forth in the chart attached as Exhibit B, I find that the following products literally infringe the following claims of the patents-in-suit, as the Court has construed these claims in its Markman Order.

a. Voltage SecureData Solution with Teradata UDFs
   "201 patent" – claims 1, 5, 19 – 25, 29 – 31

b. Voltage SecureData Solution with PL/SQL
   "201 patent" – claims 1, 5, 19 – 25, 29 - 31

c. Voltage SecureData with Tokenization
   "201 patent" – claims 1, 5, 19 – 25, 29 - 31

Highly Confidential Information: Subject to Protective Order

24
VII. DECLARATION

25. I declare under penalty of perjury that the foregoing is true and correct.

/s/ Harry Direen
Harry G. Direen Jr.
DireenTech, Inc.

Date: December 4, 2012
Exhibit A
Curriculum Vitae

Harry G. Direen Jr., Ph.D., P.E.
2750 Old North Gate Rd.
Colorado Springs, CO 80921
Phone: 719-321-6620
Email: hdireen@direentech.com

Education
Ph.D. Electronics Engineering / Control Systems
University of Colorado, 1996
Thesis: Optimization of Wavelet Basis Controllers for Nonlinear Systems with Applications to Learning Control Systems

BSEE University of California at Irvine, 1982

Professional Engineer
Registered in Colorado, USA

Experience
DireenTech Inc.
5/99 – Present

COO/Consultant
Involved in patent analysis and patent forensics work along with contracts in design and development.

The patent forensic work has involved both hardware and software product reverse engineering, which in part consists of comparing the technology in various products to inventions disclosed in a wide range of patents and the claims there-in. Products that I have reverse engineered include media players, modems, modem compression algorithms, coriolis mass flow meters, blood analysis meters and a variety of other products. I have successfully discovered and analyzed complex digital signal processing algorithms and pertinent operations within these products.

I designed and developed a 400 watt, switch-mode power supply technology, microprocessor controlled, neon light ballast for a start-up company. The ballast uses SEPIC converter coupled with a push-pull forward convertor, taking AC-line input to 10-KV output. The neon ballast has a number of unique features for which patents are being pursued. I have been responsible for all the electronic hardware design and development along with the embedded firmware design and development.

I designed and developed an automated software test system which uses a Laser-Micrometer to measure characteristics of high-precision, milling tools. The system contains motor controller for accurately placing and controlling tool position; measures tool parameters; performs significant analysis of the data, and displays the results. This is an on-going project with new features being added regularly.

I designed and developed embedded software along for an in-car camera system. The camera system captures continuous running videos of both in car and front of car locations. Video movies from before, during, and after an event are stored to a flash drive when triggered by an event such as a collision, panic button press, or door opening. The camera system contained an accelerometer, GPS, WiFi, USB, and other subsystems.

Other contracts I have been involved with include:
- High precision motor control of an optics system
- Microwave Power Amplifier for a satellite system
- FPGA design of an SPI command filter
Harry G. Direen Jr., Ph.D.

- Automated test software development for a focal plane array
- Embedded software development for high power RF amplifier

Valdez International Co
Colorado Springs, CO
5/20/2011 - present

Researcher for the Academy Center for UAS Research, Department of Electrical and Computer Engineering
Responsible for the design and development of a Heterogeneous On-board Processing System for un-manned aerial systems. This software controls autonomous, cooperative, vertical take-off and landing vehicles and fixed wing vehicles under research in the department. This is a half-time position at the Air Force Academy.

US Air Force Academy
Colorado Springs, CO

Instructor of Electrical Engineering
Temporary half time position for 1 semester. Taught ECE315, an electronic engineering course for non-engineering majors. The course covered DC electronics through radar systems.

Xpriori LLC
Colorado Springs, CO

Chief Architect
Chief architect for the Xpriori XMS native XML database, instrumental in all aspects of the architecting, design, development and coding of the database technology.

- Provide direct customer interface and support on both military classified projects and commercial projects.
- Worked with a partner company in writing and submitting an SBIR proposal for the use of the XML database product for a distributed, peer-to-peer, information management system.

NeoCore Inc.
Colorado Springs, CO

Principal/Consulting Engineer
Instrumental in the design, development and coding of NeoCore’s XML database. This included development of the entire index structure and encapsulation of NeoCore’s patented Digital Pattern Processing (DPP) into the database.

- Designed and coded a BLAST (Basic Local Alignment Search Tool) plug-in module for the XML database, to search for DNA and protein sequences.
- Design and development of network interface HW & SW. Designed and coded SW kernel and low level drivers for a firewall style network interface. HW based on QED RM7000, 64 bit RISK processor, and Intel NICs. Design of a dual 25 watt switch mode power supply for above.
- Performed finite field and statistical analysis of NeoCore’s Digital Pattern Processing (DPP) technology.

**Internal White Papers:**
- Finite Fields and Properties of the NeoCore Icon Generator, Associative Processing Unit, and Associative Memory Controller used in Digital Pattern Processing
- Couplet Hierarchy Vectors
- NeoPacket Slider with Pre-Parser and Proximity Search Engine
- Duplicate Tree Structures in DPP Virtual Associative Memories
- DPP™ Memory Management (co-author)

**Center for Computational Biology (CCB [http://www.cudenver.edu/ccb/](http://www.cudenver.edu/ccb/))**
- Established NeoCore’s connection with the UC Denver’s CCB
b. Worked with the CCB on an application for NSF grant (02-058): Self-evolving Metadata Schema for Knowledge Building in Biological Databases

c. Support of the Center of Computational Pharmacology use of NeoCore's database technology

d. Developed and gave presentations on NeoCore's XML db technology at the CCBs company showcase, CU Denver computer science dept., and UCCS biology dept.

**Recognitions**

BioT World Champion
(http://www.bio-itworld.com/champions/harry_g._direen.html)

**Advisory Boards**

2001 - 2004

Member of Advisory Board for the Center of Computational Biology
(http://www.cudenver.edu/ccb/)

Member of UCCS Advisory Board for developing undergraduate courses in the emerging area of computational biology

**Univ. of Colorado**

Colorado Springs


Taught: Engineering Probability and Statistics (Undergraduate level)

1/1999 – 5/1999

Taught: Nonlinear Adaptive Control Systems (Graduate level)

**ETO / ASTeX**

8/85 – 7/99

*Engineering Manager (3/97 – 7/99)*

Established new engineering dept/group. Technical lead and project manager on new RF amplifier for high field MRI systems. Established product specifications, schedules, budgets. Hired engineers and technicians. Primary technical interface for all U.S. and international customers.

**Senior / Principal Engineer (8/85 – 2/97)**

Principal design engineer for all embedded control for the full line of the companies RF amplifiers and RF generators. These products are used in MRI systems, semiconductor processing equipment, Laser equipment and other industrial and medical applications.

Responsibilities included hardware design of: microprocessor controls using single and multiprocessor designs, DSPs, FPGAs, A/D converters, DACs; analog signal processing, analog filters, analog feedback loops, and linear power supplies.

Responsible for all of the embedded software design, coding, and verification. Code development in assembly and C. The embedded software was responsible for: sequencing the systems on and off; user interface including parallel and serial RS-232; fault monitoring; adaptive digital feedback control loops; generator/amplifier safe operating area protection; and display interfaces.

Developed DSP based signal monitoring and processing modules. Responsible for both hardware and software development.

RF design of: low noise pre-amps; linear amplifiers; and RF signal processing. Familiar with the design of class D and class E RF amplifiers.

Taught theory of operation and service methods seminars to customers and customer service departments.

Provided product integration and support at customer sites. Often worked with customers to solve overall system level problems.
EF Johnson
4/84 – 7/85
RF / Analog Design Engineer
High power, linear RF amplifier design (150 MHz). Analog signal processing design. Semi-custom IC design of a tone generator/decoder using switch cap. filters, analog, and digital processing.

College of S. Idaho
9/84 – 12/84
Instructor
Taught course in basic electronics.

Cubic Corp.
6/82 – 3/84
Design Engineer
Design of 2 GHz, linear, solid state power amplifiers. Single ended and push-pull designs using bipolar and FET transistors. Studied linearization techniques and designed/built pre-distortion circuits. The power amplifiers were designed for use in GPS transponders on ICBMs.

Hardy Scales
7/74 – 8/81
Field Service Engineer / Technician
Provided in-house production test and field service of industrial weighing and batching equipment. Provided service in U.S., Canada, Puerto Rico, and the Virgin Islands.

US Air Force Reserves
6/73 – 6/79
Electronic Service Technician

Patents
3. US Patent No: 20030066033, Method of performing set operations on hierarchical objects
4. US Patent No: 20020069232, Method and system for generating a transform
5. US Patent No: 20020049922, Search engine system and method
6. US Patent No: 20020046205, Method of operating a hierarchical data document system having a duplicate tree structure

Publications
Harry Direen and Mark Jones, “Knowledge Management in BioInformatics” Chapter 10 of “XML and Databases”, Addison-Wesley, March 2003


Programming Languages
C, C++, C#, Python
Assembly: ARM, MIPS, Intel, TI & Motorola DSPs
Exhibit B
United States Patent 6,321,201
Data security system for a database having multiple encryption levels applicable on a data element value level

Abstract: A method and an apparatus for processing data provides protection for the data. The data is stored as encrypted data element values (DV) in records (P) in a first database (O-DB), each data element value being linked to a corresponding data element type (DT). In a second database (IAM-DB), a data element protection catalogue (DC) is stored, which for each individual data element type (DT) contains one or more protection attributes stating processing rules for data element values (DV), which in the first database (O-DB) are linked to the individual data element type (DT). In each user-initiated measure which aims at processing a given data element value (DV) in the first database (O-DB), a calling is initially sent to the data element protection catalogue for collecting the protection attribute/attributes associated with the corresponding data element types. The user's processing of the given data element value is controlled in conformity with the collected protection attribute/attributes.

Inventors: Dahl, Ulf

Assignee: Protegrity Corporation or Anonymity Protection in Sweden AB (Gothenburg, SE)

Appl. No.: 09/027,585
Filed: February 23, 1998
Granted: November 20, 2001

Current US Class: 705/51; 707/999.009
Current International Class: G06F 1/00 (20060101); G06F 21/00 (20060101); G06F 017/60 ()

Field of Search: 380/4,3 707/104,1,9 705/1,50,51,54

Asserted Claim: Claim 1, Claim 5, Claims 19 – 25, Claims 29 - 31
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A method for processing of data that is to be protected, comprising:</td>
<td>See Abstract</td>
<td>Voltage SecureData Products implement a method for processing of data that is to be protected. This method is comprised of the following steps:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>As seen in Figure 1, Voltage SecureData is designed to protect data stored in databases. Figure 2 shows the components that make up the Voltage SecureData products. Figure 20 shows some of the software components that make up the SecureData product.</td>
</tr>
</tbody>
</table>

![Overview of Voltage SecureData](image)

*Figure 1 Voltage SecureData Protecting Data*¹

Voltage SecureData Software

The Voltage SecureData software offers comprehensive data protection options, including support for Voltage IBE, Voltage PPE, tokenization, embedded key information, and other traditional protection methods such as AES. These options ensure that all data can be protected, regardless of data structure or application format requirements.

Voltage SecureData software can include the following components:

- Voltage SecureData Key Management Server—Centralized Voltage key management.
- Voltage SecureData Management Console—Centralized configuration and reporting across the entire Voltage SecureData solution.
- Voltage SecureData Web Services Server—Web Service data protection option for SOA environments.
- Voltage SecureData Simple API—A simplified interface on top of the Voltage Encryption Toolkit that is intended to make incorporating the protection/access functionality directly into existing applications.
- Voltage Encryption Toolkit—High-performance low-level IBE Elliptic Curve Cryptography toolkit with work with standard cryptographic libraries such as AES, SHA256, RSA, etc. The Toolkit interfaces directly with the Voltage SecureData Key Management Server.
- Voltage POS SDK—A light-weight C SDK with a subset of low-level IBE Elliptic Curve Cryptography toolkit protection functions specifically designed to be embedded directly in Point of Sales (POS) devices or software-based ECR systems.
- Voltage Host SDK—a subset of low-level IBE Elliptic Curve Cryptography toolkit data protection functions specifically designed to be incorporated into a payments host. These APIs take data and keys protected with the POS API and access and then reprotect the data before it enters the payment providers database.
- Voltage SecureData for z/OS—Provides the full Simple API functionality on z/OS, callable from LE or any other compiled language. Includes z/PPF, an advanced bulk data protection tool for protecting flat files or DB2 data as it is unloaded via IBM DB2 HPU.
- Voltage SecureData Web—Encryption of PAN and CVV data for web-based purchases.

Figure 20 Voltage SecureData Software (ATT001032 - 3)
Figure 21 SecureData Protection Attributes for a Data Type

Table 1 SecureData Database Table (FormatSpec) for Data Types
(This is a partial table, not all column data is shown)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Data type name, used to reference this data type</td>
</tr>
<tr>
<td>formatType</td>
<td>VAR String, FIX String, Credit Card number, US SSN</td>
</tr>
<tr>
<td>inputType</td>
<td>String, Oracle Date, Number</td>
</tr>
<tr>
<td>unicode</td>
<td>Boolean to determine if string type is unicode</td>
</tr>
<tr>
<td>ignoreOutOfRange</td>
<td></td>
</tr>
<tr>
<td>characterSet</td>
<td>Input alphabet</td>
</tr>
<tr>
<td>outputCharacterSet</td>
<td></td>
</tr>
<tr>
<td>maxKeyNumCount</td>
<td>Used for maximum key rotation number in eFPE</td>
</tr>
<tr>
<td>formatString</td>
<td></td>
</tr>
<tr>
<td>minValue</td>
<td>Minimum value for a number</td>
</tr>
<tr>
<td>maxValue</td>
<td>Maximum value for a number</td>
</tr>
<tr>
<td>retain_leading</td>
<td>For credit cards the number of leading numbers to retain</td>
</tr>
<tr>
<td>retain_trailing</td>
<td>For credit cards the number of trailing numbers to retain</td>
</tr>
<tr>
<td>lookup_db</td>
<td>Database for Tokenization</td>
</tr>
<tr>
<td>lookup_table</td>
<td>Database Table for Tokenization</td>
</tr>
<tr>
<td>lookup_identity</td>
<td>Identity used for Tokenization</td>
</tr>
<tr>
<td>other</td>
<td>There are a variety of additional column definitions</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 5. A method as claimed in claim 1, wherein the protection attribute/attributes of the data element types comprise attributes stating rules for encryption of the corresponding data element values in the first database (O-DB). | **Construction:**  
I construe the phrase "in the first database" to have its plain meaning.  

**Data element value:** "a value which in a given record specifies a data element type."  

**Column 8, line 2**  
"If the descriptive information DI at issue is to be stored as a data element value DV associated with the specific data element type DT1 in the data element protection catalogue, the protection attribute "5" registered in the data element protection catalogue is collected automatically in this case. The descriptive information DI at issue will thus, automatically and compellingly, be encrypted with the strength "5" for generating an encrypted data element value DV" | See Claim 1.  
In the method implemented by the Voltage SecureData Product the protection attribute/attributes of the data element types comprise attributes stating rules for encryption of the corresponding data element values in the first database. This is evidenced as follows:  

In Figure 21 the “Data Protection Type” is an attribute stating rules for encryption of the corresponding data element values in the first database. In Figure 21 the Data Protection Type is FPE (Format Preserving Encryption) with embedded key number. This encryption rule states that the key number must be embedded into the encrypted value.  

Figure 22 shows that the encoded identity used to obtain an encryption key from the SecureData Server uses an “Algorithm” protection attribute. This attribute states rules for encryption of the data element values. |
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name</td>
<td>The common name field is the foundation of the identity. Currently, the only type of common name supported by Voltage SecureData is in the form of an email address, which is case-insensitive.</td>
</tr>
<tr>
<td>Identitytimes</td>
<td>When the time field is used with symmetric protection, the date is a not before time and acts as another field that defines the key; changing the time is a way to roll over keys.</td>
</tr>
<tr>
<td>District</td>
<td>The domain name plus serial number. For more information about districts, see &quot;Understanding Districts&quot; on page 2-7.</td>
</tr>
<tr>
<td>Application</td>
<td>Specifies what type of data Voltage SecureData is protecting: disk data, database fields, etc. If not specified, the default is email.</td>
</tr>
<tr>
<td>Algorithm</td>
<td>An identifier of the algorithm used to protect the data. In this way, if the rest of the identity information (email address, key size, etc.) is the same, the key is still different for different protection algorithms.</td>
</tr>
<tr>
<td>Key Size</td>
<td>If the algorithm identifier specifies a key size, the key size field could be Triple-DES, AES-128-CBC, etc.</td>
</tr>
</tbody>
</table>

Figure 22 Encryption Algorithm Attributes"
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>19. The method as set forth in Claim 1, wherein the data element protection catalogue (DPC) containing each individual data element type (DT), comprises storing each individual data element type (DT) as data element type data in the data element protection catalogue (DPC).</td>
<td>Column 2, Line 46 “The inventive method is characterised by the following further measures: storing in a second database (IAM-DB) a data element protection catalogue, which for each individual data element type contains one or more protection attributes stating processing rules for data element values, which in the first database are linked to the individual data element type,”</td>
<td>See claim 1.</td>
</tr>
</tbody>
</table>

In the method implemented by the Voltage SecureData Product, the data element protection catalogue containing each individual data element type, comprises storing each catalogue. This is evidenced as follows:

Figure 23 shows a SecureData Management Console for establishing a “Salary” data type. Part of the salary data type is Minimum and Maximum salary values. The Minimum and Maximum salary values are data element data type data that are stored in the data protection catalog within the SecureData Server’s database. Table 1 has minValue and maxValue columns for storing this data.
Figure 23 SecureData Management Console for Establishing a Salary Data Type\textsuperscript{30}

\textsuperscript{30} Voltage SecureData Administrator Guide, Version 5.0, page 77 ATT001192
| US Patent 6,321,201  
Claim 20 Limitations | US Patent 6,321,201  
Claim Construction and 
Specification Support | Voltage’s SecureData Database System Product Elements |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20. The method as set forth in claim 19, wherein the data element protection catalogue (DPC) containing the one or more protection attributes stating processing rules for data element values (DV), comprises storing the one or more protection attributes stating processing rules for data element values (DV) as data element value data in the data element protection catalogue (DPC).</td>
<td>Construction: &quot;A data element protection catalogue (DPC), which contains -- in a location other than a column heading of a table -- each individual data element type (DT) and one or more protection attributes stating processing rules for data element values associated with the data element types, where there is a direct mapping between data element types and their respective protection attributes.&quot;</td>
<td>See Claim 19.</td>
</tr>
</tbody>
</table>
| **Data element value:** “a value which in a given record specifies a data element type.” | **Column 3, Line 24**  
"Protection attribute indicating rules of processing" may concern: data stored in the data element protection catalogue and | In the method implemented by the Voltage SecureData Product the data element protection catalogue (DPC) containing the one or more protection attributes stating processing rules for data element values, comprises storing one or more protection attributes stating processing rules for data element values as data element value data in the data element protection catalogue. This is evidenced as follows: |
| When a security administrator sets up a data type through the SecureData Management Console, a number of protection attribute values are selected, Figure 24. It the example shown in Figure 24, the security administrator selected a “Data Protection Type” of eFPE. A “Key Rotation Group” was selected and an “Output Alphabet” has been established. The values in the protection attributes represent data element value data that must be stored in the data protection catalog within the SecureData Server’s database. In addition to these protection attributes, Format Mask Settings may also be established for the data type. The Format Mask Settings represent additional protection attributes stating rules for data element values. The Format Mask Settings protection attributes will be stored as data element value data in the data protection catalog. |
|----------------------------------------|---------------------------------------------------------------|--------------------------------------------------------|
| providing complete information on the rule or rules applying to the processing of the corresponding data element, and/or data stored in the data element protection catalogue and requiring additional callings to information stored in some other place, which, optionally in combination with the protection attributes, states the processing rules involved.” |
Figure 24 SecureData Protection Attribute Value Data\textsuperscript{31}

\textsuperscript{31} Voltage SecureData Administrator Guide, Version 5.0, page 66 ATT001181
| Case 3:12-cr-00268-RNC   Document 4-4    Filed 12/19/12   Page 11 of 30 |

<table>
<thead>
<tr>
<th><strong>US Patent 6,321,201</strong></th>
<th><strong>Claim Construction and Specification Support</strong></th>
<th><strong>Voltage's SecureData Database System Product Elements</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Claim 21 Limitations</strong></td>
<td><strong>Construction:</strong> Data element value: &quot;a value which in a given record specifies a data element type.&quot;</td>
<td><strong>See Claim 20.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Column 2, Line 46</strong> &quot;The inventive method is characterised by the following further measures: storing in a second database (IAM-DB) a data element protection catalogue, which for each individual data element type contains one or more protection attributes stating processing rules for data element values, which in the first database are linked to the individual data element type.&quot;</td>
<td><strong>In the method implemented by the Voltage SecureData Product the storing of the data element type data and the data element value data in the data element protection catalogue, comprises storing data element type data and its corresponding data element value data in rows in the data element protection catalogue. This is evidenced as follows.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Column 4, Line 45</strong> &quot;Statement of what &quot;strength&quot; or &quot;level&quot; (for instance none, 1, 2 ... ) of encryption is to be used for</td>
<td><strong>Table 1 shows the structure of a portion of SecureData's data element protection catalog. Columns formatType, minValue, maxValue, unicode, and others store data element type data. Columns outputCharacterSet, retain_leading, retain_trailing, and others contain data element value data which state processing rules for data element values. The data element type data and the corresponding data element value data is stored in rows in the data element protection catalog.</strong></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>storing the corresponding data element values in the database. Different data element values within one and the same record may thus be encrypted with mutually different strength.”</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Column 4, Line 60</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“Statement of sorting-out rules for the corresponding data element values, for instance, statement of method and time for automatic removal of the corresponding data element values from the database.”</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------------------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>22. The method as set forth in claim 21, wherein the storing of the data element type data and the data element value data in the data element protection catalogue (DPC), comprises storing data element type data and its corresponding data element value data in different rows in the data element protection catalogue (DPC).</td>
<td>Construction: Data element value: “a value which in a given record specifies a data element type.”</td>
<td>See Claim 21.</td>
</tr>
<tr>
<td></td>
<td>Column 4, Line 45 “Statement of what &quot;strength&quot; or &quot;level&quot; (for instance none, 1, 2 ...) of encryption is to be used for storing the corresponding data element values in the database. Different data element values within one and the same record may thus be encrypted with mutually different strength.”</td>
<td>In the method implemented by the Voltage SecureData Product the storing of the data element type data and the data element value data in the data element protection catalogue, comprises storing data element type data and its corresponding data element value data in different rows in the data element protection catalogue. This is evidenced as follows.</td>
</tr>
<tr>
<td></td>
<td>Column4, Line 60 “Statement of sorting-out rules for the corresponding data element values, for instance, statement of method and time for automatic removal of the corresponding data element values from the database.”</td>
<td>Table 1 shows the structure of a portion of SecureData’s data element protection catalog. Each row of this table contains the data element type data and the corresponding data element value data for a data type. Each data type stores the data element type data and the corresponding data element value data in a different row in the data element protection catalog.</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Column 3, Line 8</td>
<td>&quot;Record&quot; concerns a number of data element values which belong together and which are linked to the respective data element types, optionally also including a record identifier, by means of which the record can be identified.</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
</tbody>
</table>
| 23. The method as set forth in claim 22, wherein the data element type data and its corresponding data element value data are stored in respective fields in the rows. | **Construction:**  
Data element value: “a value which in a given record specifies a data element type.”  
**Column 4, Line 45**  
“Statement of what "strength" or "level" (for instance none, 1, 2 ...) of encryption is to be used for storing the corresponding data element values in the database. Different data element values within one and the same record may thus be encrypted with mutually different strength.”  
**Column 4, Line 60**  
“Statement of sorting-out rules for the corresponding data element values, for instance, statement of method and time for automatic removal of the corresponding data element values from the database.” | See Claim 22.  
In the method implemented by the Voltage SecureData Product, the data element type data and its corresponding data element value data are stored in respective fields in the rows. This is evidenced as follows:  
Table 1 shows the structure of a portion of SecureData’s data element protection catalog. The data element type data and the corresponding data element value data are stored in their respective fields in the rows. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage's SecureData Database System Product Elements</td>
<td></td>
</tr>
</tbody>
</table>

Column 3, Line 8
"Record" concerns a number of data element values which belong together and which are linked to the respective data element types, optionally also including a record identifier, by means of which the record can be identified."
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>24. The method as set forth in claim 23, wherein each row comprises one field containing one of the data element type data and other fields containing its one or more protection attributes.</td>
<td><strong>Column 3, Line 8</strong> “Record” concerns a number of data element values which belong together and which are linked to the respective data element types, optionally also including a record identifier, by means of which the record can be identified.”</td>
<td>See Claim 23.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In the method implemented by the Voltage SecureData Product, each row comprises one field containing one of the data element type data and other fields containing its one or more protection attributes. This is evidenced as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Table 1 shows the structure of a portion of SecureData’s data element protection catalog. The <code>formatType</code> field contains data element type data. The <code>outputCharacterSet</code>, <code>retain_leading</code>, <code>retain_trailing</code>, and other fields contain protection attributes for the <code>formatType</code> data type.</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
</tbody>
</table>
| 25. The method as set forth in Claim 24, further comprises adding a new data element type data (DT) in a new row in the data protection catalogue (DPC) and adding new data element value data of one or more protection attributes stating processing rules for that data element value (DV) associated with the new data element type (DT) as data element type data in other new fields of the new row and linking at least one of the data element value in the first database to the new data element type (DT) in the data protection catalogue (DPC). | Construction: I construe the phrase "in the first database" to have its plain meaning. "A data element protection catalogue (DPC), which contains -- in a location other than a column heading of a table -- each individual data element type (DT) and one or more protection attributes stating processing rules for data element values associated with the data element types, where there is a direct mapping between data element types and their respective protection attributes." Data element value: “a value which in a given record specifies a data element type.” | See Claim 24  
The method implemented by the Voltage SecureData Product comprises adding a new data element type data in a new field in a new row in the data protection catalogue and adding new data element value data of one or more protection attributes stating processing rules type data in other new fields of the new row and linking at least one of the data element value in the first database to the new data element type in the data protection catalogue. This is evidenced as follows:  

Figure 25 shows a SecureData Management Console for adding a Salary data type to the data protection catalog. Each new data type established through the SecureData Management Console adds a new row in Table 1. The new row will contain the data element type data and the corresponding data element value data as set by the security administrator through the SecureData Management Console. The data element type data and the corresponding data element value data will be added to the fields in the new row in the data element protection catalog.
Figure 25 SecureData Management Console for Establishing a Salary Data Type

32 Voltage SecureData Administrator Guide, Version 5.0, page 77 ATT001192
|---------------------|---------------------|---------------------------------------------------|
| 29. The method as set forth in claim 1, wherein the database protection catalogue is inaccessible to a user. | **Construction:**
Plain meaning of the word “user” which I do not take to be limited to persons.

**Column 6, Line 2**
“This database IAM-DB is preferably physically separated from the other O-DB and is inaccessible to the user.” | In the method implemented by the Voltage SecureData Product, the database protection catalogue is inaccessible to a user. This is evidenced as follows:

The SecureData Server’s database is only available to an administrator of the SecureData Server so it is not available to user accessing data in the first database. The configuration files, clientPolicy.xml and vsconfig.txt, published by the SecureData Server are stored outside of the first database and as such are not accessible to the user of the first database. Therefore the data protection catalog is inaccessible to a user.
|----------------------------------------|---------------------------------------------------------------|----------------------------------------------------|
| 30. The method as set forth in claim 1, wherein the database protection catalogue is physically separate from the first database. | **Column 6, Line 2**
“This database IAM-DB is preferably physically separated from the other O-DB and is inaccessible to the user.” | See Claim 1. |

In the method implemented by the Voltage SecureData Product, the database protection catalogue is physically separate from the first database. This is evidenced as follows:

The SecureData Server’s database is contained within the SecureData Server which is physically separate from the first database.

The VibeSimple PL/SQL Package used with Oracle databases makes calls to the SecureData SOA Web Server which is physically separate from the Oracle database. The clientPolicy.xml configuration file published by the SecureData Server is stored on the SecureData SOA Web Server and not at the Oracle database. Therefore the data protection catalog is physically separate from the first database.

SecureData Tokenization also uses the SecureData SOA Web Server (Figure 27) which is physically separate from the first database. Therefore the data protection catalog is physically separate from the first database when using the SecureData Tokenization solution.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>There are two different ways to call the Voltage SecureData Encryption Web Services from PL/SQL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using the <em>vibesoap.sql</em> PL/SQL package that uses UTL_HTTP to connect to the Voltage SecureData SOA Web Services. Use this package when you are connecting to SecureData using HTTP or HTTPS with Oracle version 10g or later.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using the <em>vibesoaj.sql</em> PL/SQL package that connects to the Voltage SecureData SOA Web Services using Java stored procedures. Use this package when you are accessing SecureData via HTTPS from an Oracle 9i server.</td>
</tr>
</tbody>
</table>

**Figure 26 SecureData PL/SQL uses SOA Web Services**

---

**Note:** Tokenization is only available through the Voltage SecureData Encryption Web Services interface. This feature is not available through the Voltage SecureData CL or the Simple API.

**Figure 27 SecureData Tokenization uses Web Service Interface**

---

33 Voltage SecureData Developer Guide, Version 4.0.1a, page 4-1 (ATT000358)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31. The method as set forth in claim 1, wherein the data-base protection catalogue is physically separate from the first database and is inaccessible to a user.</td>
<td><strong>Construction:</strong> Plain meaning of the word “user” which I do not take to be limited to persons. <strong>Column 6, Line 2</strong> “This database IAM-DB is preferably physically separated from the other O-DB and is inaccessible to the user.”</td>
<td>See Claim 1. In the method implemented by the Voltage SecureData Product, the data-base protection catalogue is physically separate from the first database and is inaccessible to a user. This is evidenced as follows: As seen in claims 29 and 30, the data protection catalog is both physically separate and inaccessible to a user.</td>
</tr>
<tr>
<td>File Name</td>
<td>File anDocumentl</td>
<td>Dale</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Platforms May 28 2009 final</td>
<td>Pl at r l r ata Corl Platform Pt pprr Partn rl rship P ith om orr</td>
<td>Mayt09</td>
</tr>
<tr>
<td>Pr aorahitl att rl a3.0.2</td>
<td>Poltaol Pl at r l r ata oarahitl att rl Gt idl</td>
<td>PantP0</td>
</tr>
<tr>
<td>Pr at l staPraatial sar l ahNotl</td>
<td>Poltaol Pl at r l rl ata r l vl lopl rl t l st Praatial s r l ahniaal Notl</td>
<td>Pl pt09</td>
</tr>
<tr>
<td>Pr ar l vl lopl ra3.0.2</td>
<td>Poltaol Pl at r l r ata r l vl lopl r Gt idl</td>
<td>PantP0</td>
</tr>
<tr>
<td>Pr ar l vl lopl ra l 0 Pa</td>
<td>Poltaol Pl at r l r ata r l vl lopl r Gt idl</td>
<td>PantP0</td>
</tr>
<tr>
<td>Pr atf Par l vl lopl ra2,P.0</td>
<td>Poltaol Pl at r l r ata for trf Pr l vl lopl r Gt idl</td>
<td>Pl atP0</td>
</tr>
<tr>
<td>Pl ssionPtCsharp</td>
<td>Poltaol Pl at r l r ata r rainino</td>
<td>PantP0</td>
</tr>
<tr>
<td>Pl ssion2tJava</td>
<td>Poltaol Pl at r l r ata r raininoJava oPc</td>
<td>PantP0</td>
</tr>
<tr>
<td>Pl ssion3tPlP</td>
<td>Poltaol Pl at r l r ata r rainino for trf Pl</td>
<td>PantP0</td>
</tr>
<tr>
<td>Poltaol Pl at r l r ata r MainframPl</td>
<td>Poltaol Pl at r l r ata r MainframPl oarahitl att r l</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol Pl at r l r ata r PoP dtl oration and r l mo slidl s</td>
<td>Poltaol Pl at r l r ata r PoP dtl oration and r l mo slidl s</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol Pl at r l r ata r l vl lopl r r rainino Pl ptrl mal r 2008</td>
<td>Poltaol Pl at r l r ata r l vl lopl r r rainino Pl ptrl mal r 2008</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol Pl at r l r ata r l ah Notl t MainframPl Pl rformanal Gt idanal tf PPPt2</td>
<td>Poltaol Pl at r l r ata r l ah Notl t MainframPl Pl rformanal Gt idanal tf PPPt2</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol r oolyit Pimpil oPd3.P.P r Pror Codl s</td>
<td>Poltaol r oolyit Pimpil oPd3.P.P r Pror Codl s</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol aPNaryptionar oolyitaPimpil oPd3.8.x</td>
<td>Poltaol aPNaryptionar oolyitaPimpil oPd3.8.x</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol aPI at r l r atal .P.PaRI INotl</td>
<td>Poltaol aPI at r l r atal .P.PaRI INotl</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol aPI at r l r ataa0.0ar ataaasi aPI rformanal atl ahnotl</td>
<td>Poltaol aPI at r l r ataa0.0ar ataaasi aPI rformanal atl ahnotl</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Poltaol aPI at r l r ataa0a0aRI INotl</td>
<td>Poltaol aPI at r l r ataa0a0aRI INotl</td>
<td>no datl listl d</td>
</tr>
<tr>
<td>Voltage_SecureData_CL_User_GuideV</td>
<td>Voltage_SecureData_Voltage_SecureData_CmdLineUserGuideV</td>
<td>voltagV</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Voltage_SecureData_Ceradata_aiai2_ReiVoteV</td>
<td>Voltage_SecureData_Vbr_Ceradata_aiai2_Release_VotesV</td>
<td>DectaaV</td>
</tr>
<tr>
<td>Voltage_SecureDataCL_2iV_ReiVoteV</td>
<td>Voltage_SecureData_Vcl_W version_VBr_Vig_ReleaseVotesV</td>
<td>Vol_tgV</td>
</tr>
<tr>
<td>Voltage_SecureDataSimmenPI_ViaVigd_Januar12ga2_ReiVoteV</td>
<td>Voltage_SecureData_VsimmenVPI_Version_ViaVigd_ReleaseWotesV</td>
<td>Janta2V</td>
</tr>
<tr>
<td>VoltageSD_nmmliance_Lig_nrrGitectureV</td>
<td>Voltage_SecureData_VrrGitectureVGuideV</td>
<td>V-ttaaV</td>
</tr>
<tr>
<td>VoltageSD_nmmliance_Lig_n anagementConso le_nadminV</td>
<td>Voltage_SecureData_VnministratorVGuideV</td>
<td>V-ttaaV</td>
</tr>
<tr>
<td>VoltageSD_nmmliance_Lig_n el Seriices_Del elomerV</td>
<td>Voltage_SecureData_VDel_elomerVGuideV</td>
<td>V-ttaaV</td>
</tr>
<tr>
<td>VoltageSD_nmmliance_UmdategVvgaa_InstallV</td>
<td>Voltage_SecureData_VumdateVinstallation\InstructionsV</td>
<td>V-ttaaV</td>
</tr>
<tr>
<td>VoltageSD_SimmenPI_ViaV_Del elomerV</td>
<td>Voltage_SecureData_VsimmenVPI_Del elomerVGuideV</td>
<td>nmrtaaV</td>
</tr>
<tr>
<td>VoltageSD_Ceradata_aiai2_Del elomerV</td>
<td>Voltage_SecureData_Vbr_Ceradata_Del elomerVGuideV</td>
<td>DectaaV</td>
</tr>
<tr>
<td>CoRCiglCnCIV\V\PnGoS\br\VoGn\eald\Demi\VtaVta2V</td>
<td>CertVCate\V\Demosition\Vbr\V2\V\V\ on LDV</td>
<td>vsenta2V</td>
</tr>
<tr>
<td>CoRCiglCnCIV\V\PnGoS\br\VimotGI \V\Gommmson\Dem\vt2Vta2V</td>
<td>CertVCate\V\Demosition\Vbr\VimotGI \V\GommmsonV</td>
<td>nugi\V2ga2V</td>
</tr>
<tr>
<td>tGommmson_VimotGI \V\oo21aICV\V</td>
<td>United\States\District\Court\V\CeRas\Sui\nen\a\to\V\est\V\V\Demosition\V\V\il\V\ction\V</td>
<td>Voa22ga2V</td>
</tr>
<tr>
<td>tGommmson_VimotGI \V\oo21aICV2V</td>
<td>United\States\District\Court\V\CeRas\Sui\nen\a\to\V\est\V\V\Demosition\V\V\il\V\ction\V</td>
<td>Voa22ga2V</td>
</tr>
<tr>
<td>tGommmson_VimotGI \V\oo21aICV\V</td>
<td>United\States\District\Court\V\CeRas\Sui\nen\a\to\V\est\V\V\Demosition\V\V\il\V\ction\V</td>
<td>VoV22ga2V</td>
</tr>
<tr>
<td>Thompson, Timothy t oan</td>
<td>t oapsT, Tci rTDtpt DTvT\text{\textcircled{ap}} Tr Gi idT</td>
<td>tpntps</td>
</tr>
<tr>
<td>thompson, timothy t h\textordf\textordf{J}a\textordf t h\textordf{J}a\textordf\textordf{H}p\textordf{H}d,</td>
<td>t nITd, tptTs Distrcit Coi rt of ConnTctici t</td>
<td>t hti npt</td>
</tr>
<tr>
<td>Tpad, tohn \text{\textcircled{O}}n</td>
<td>t nITd, tptTs Distrcit Coi rt of ConnTctici t</td>
<td>ti \textcircled{O}pt</td>
</tr>
<tr>
<td>Tpad, tohn t t or</td>
<td>t nITd, tptTs Distrcit Coi rt of ConnTctici t</td>
<td>ti \textcircled{O} t h t t spt</td>
</tr>
<tr>
<td>mosTr, r\text{\textcircled{O}}sTph sp</td>
<td>, i rsTrTs to\textcircled{T}spTph t \text{\textcircled{O}}d osTr</td>
<td>t t tti app</td>
</tr>
<tr>
<td>mosTr, r\text{\textcircled{O}}sTph st</td>
<td>tist of D Tdictioncns for tosTph t \text{\textcircled{O}}d osTr</td>
<td>ti nT t t spt</td>
</tr>
<tr>
<td>mosTr, r\text{\textcircled{O}}sTph sT</td>
<td>tosTph t \text{\textcircled{O}}d osTr's D Tdicipomsi rpnCT</td>
<td>ndTntificpion</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n sp</td>
<td>t nITd, tptTs Distrcit Coi rt of ConnTctici t</td>
<td>ti \textcircled{O}y ph t spt</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n st</td>
<td>t oapsT, Tci rTDtpt TrchitTctti rT Gi idT</td>
<td>t cttpp</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n st</td>
<td>t oapsT, Tci rTDtpt T\text{\textcircled{O}}m\text{\textcircled{O}}n\text{\textcircled{O}}r\text{\textcircled{O}}r\text{\textcircled{T}}r\text{\textcircled{T}}r\text{\textcircled{T}}r\text{\textcircled{T}}r\text{\textcircled{T}}T</td>
<td>t cttpp</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n s,</td>
<td>t oapsT, Tci rTDtpt o, D, i ppa\text{\textcircled{O}}mTnt</td>
<td>t cttpp</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n so</td>
<td>t oapsT, Tci rTDtpt TovTntlption DTP\text{\textcircled{O}}ymTnt</td>
<td>t ohibit o sh/ph/pt</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n s6</td>
<td>t oapsT mtTsrption for t oapsT, Tci rTDtpt, T t</td>
<td>t ohibit 6 sh/ph/pt</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n sh</td>
<td>DptpbpsT TrissTrs so\text{\textcircled{O}}nt</td>
<td>t ohibit h sh/ph/pT</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n sh</td>
<td>, Tci rTDtpt Tp\text{\textcircled{O}}p\text{\textcircled{O}}p\text{\textcircled{O}}p\text{\textcircled{O}}p\text{\textcircled{O}}p\text{\textcircled{T}}C tC</td>
<td>t ohibit h sh/ph/pT</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n st</td>
<td>t oapsT, Tci rTDtpt \text{\textcircled{O}}m\text{\textcircled{O}}r\text{\textcircled{O}}s\text{\textcircled{O}}p\text{\textcircled{T}}d t nTnT t sTr Gi idT</td>
<td>t ohibit t sh/ph/pT</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n ps</td>
<td>Ds, pTy GTnTrptor DT\text{\textcircled{O}}s\text{\textcircled{O}}n, pTctificpion</td>
<td>t ohibit ps sh/ph/pT</td>
</tr>
<tr>
<td>Tpad, r\text{\textcircled{O}}h\text{\textcircled{O}}n h\text{\textcircled{O}}p\text{\textcircled{O}}h\text{\textcircled{O}}p\text{\textcircled{O}}h\text{\textordf{P}}t</td>
<td>CTrtificpT of DTposion for tt o D t t TtD</td>
<td>h/ph/t spt</td>
</tr>
</tbody>
</table>
Exhibit D
July 31, 2012

Mr. Stefan V. Stein
Gray-Robinson, P.A.
201 North Franklin Street, Suite 2200
Tampa, Florida 33602

Sub:      TAEUS Quote 12-GR001-000114 for Expert Witness and Technical Support
Ref:     Your 6/7/12 Request to find experts in relational database and encryption

Dear Mr. Stein,

TAEUS International Corporation (TAEUS) is pleased to provide you this quotation for the technical expertise, support, and management of Dr. Harry Direen on behalf of Protegrity Corporation in its current litigation efforts.

This project is best handled on a time-and-materials basis at the following rates:

- Subject Matter Expert .......................................................... $275.00/hour
- Expert Report Preparation .................................................. $300.00/hour
- Deposition/Testifying Time/On-Site .................................... $375.00/hour
- Travel Time ........................................................................ $200.00/hour

- TAEUS Engineering Support Services .................................... $350.00/hour
- TAEUS Admin Support Services (Research/Publishing/Graphics) .... $300.00/hour

TAEUS reserves the right to revisit hourly rates on an annual basis. Services outside the US or Canada carry an additional charge of $50.00 per hour. Travel costs directly related to this project (i.e., food, lodging, airfare) are billed at our cost.

A nonrefundable retainer of $10,000 will be invoiced upon your authorization to proceed, and is due upon receipt. Subsequent invoices will reflect the actual time used on the project including conference calls. Invoices for work completed are compiled and submitted semi-monthly (and weekly when appropriate).

An approved signature on the bottom of this letter and return fax of the same to +1-719-632-5175 will serve as your authorization to proceed and agreement to TAEUS terms included on the following page. This quotation is valid for 30 days.

TAEUS International Corporation
1155 Kelly Johnson Boulevard, Suite 400, Colorado Springs, Colorado 80920 USA
tel: +1.719.325.5000 | fax: +1.719.632.5175 | www.taeus.com
Confidential | Page 1 of 3
Thank you for the opportunity to be of service to you, Gray-Robinson, P.A. and Protegrity Corporation.

Sincerely,

Arthur Nutter  
President and Chief Executive Officer

Authorized Signature:  
[Signature]

Date: 8/9/2012

Please list the contact information for the person in Accounts Payable that should receive invoices.

**Protegrity Corporation**

Name
C/O Protegrity USA, Inc

Address 1
5 HIGH RIDGE PARK

Address 2

City  
STAMFORD, CT 06905

Country  
USA

ATTN: ANDREA AHLQUIST

Phone

email

fax

State or Province