METHODS OF DATA PROTECTION
1. Introduction

Data protection comes in many forms. Conventional methods include good data governance, network firewalls, Intrusion Prevention Systems (IPS), Identity Management (IDM), semantic layer row- and column-level security (RLS/CLS), Role-Based Access Controls (RBAC), activity monitoring, encryption, tokenization, masking, obfuscation and many more. This whitepaper will focus on a data-centric approach that provides an essential additional layer of security for protecting sensitive data today.

Data-centric protection is focused on the data itself, protecting it wherever it is stored (at rest), moved or copied (in transit) and accessed or used (in use). Protection can be coarse-grained (protecting entire disk volumes, directory contents, or files) or fine-grained (protecting the individual values in fields or columns). Data-centric protection can be enforced using a variety of methods, depending on the particular use case or data repository.

2. Definitions

Many of the terms used to describe data protection methods are misused creating confusion in the marketplace. The following provides a lexicon for the different data protection methods and how these terms are used in this document.

2.1. Dynamic Data Masking (DDM)

DDM does not alter the cleartext data at rest. Views are created that mask all or part(s) of the data when displayed to the user. While many systems require this added security, most databases do this natively within views or other means. The term “masking” is often misused to describe almost any data protection process.

<table>
<thead>
<tr>
<th>Cleartext vs. Dynamically Masked Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>381-58-6294</td>
</tr>
</tbody>
</table>

2.2. Static Data Masking (SDM) / Data Obfuscation

SDM alters the cleartext data to create values that often look much like production data, but contain no real data, and is used frequently in development or test environments. Obfuscated data cannot be reversed to return to the original data.

One-way hash algorithms use encryption technology to achieve this in a consistent way, where the message digest (hash value) will be binary data which is irreversible and cannot be stored using the original data type. Most data obfuscation tools generate output that looks like original data and can continue to be stored using the same data type and character set.
2.3. Data Encryption

Encryption technology utilizes mathematical algorithms and cryptographic keys to alter data into binary ciphertext. It is reversible only using the correct key with the algorithm. There are many forms of data encryption, various key strengths and other options. Encrypted (ciphertext) output is binary data and looks nothing like the original cleartext and normally requires changing the data type for the field.

2.4. Data Tokenization

Tokenization has existed since before there were computers. In its most basic form, it is simply substituting a randomly generated value (token) for a cleartext value and keeping a lookup table (token vault) in a secure place, which maps the cleartext value to the corresponding token. The token data type and length typically remains the same as the cleartext, and the token lookup table becomes the “key,” allowing the cleartext value to be retrieved from the token. However, as the tokenized data set grows and IT infrastructure becomes more complex, these dynamic token lookup tables quickly become unmanageable.

Protegrity provides an elegant and efficient Vaultless Tokenization solution (PVT) that uses small, static token tables to create unique, random token values without the need for a dynamic token lookup table. The result is a highly scalable, flexible and powerful protection method for structured or semi-structured data.

2.5. Format-Preserving Encryption (FPE)

Format Preserving Encryption provides some benefits of both encryption (also uses standards-based mathematical algorithm) and tokenization (also can preserve same data type as original cleartext data).
However, these combined benefits come at a cost. FPE requires the same CPU cycles to encrypt, then additional processing to convert the binary ciphertext into the same data type as the original and avoid "collisions" (the same output for two different input values) resulting from converting a larger binary field into a smaller alpha, numeric, or alphanumeric data type.

Limited support for extreme field lengths and various data types are also negative factors in the use of FPE. FPE may be referred to by other names such as Data-Type-Preserving Encryption (DTPE).

2.6. De-Identification / Anonymization

De-identification is the more general term or “catch-all” phrase for “anonymizing” data to render the actual person the records are associated with nearly impossible to identify using the remainder of the cleartext data. For example, a customer table where the Name, Address, Phone, DOB and Email fields are in some way masked, obfuscated, encrypted or tokenized while the remainder of the fields like State, Country, Income, Type of Customer, and other fields are left in the clear.

<table>
<thead>
<tr>
<th>Original vs. De-Identified Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST</td>
</tr>
<tr>
<td>John</td>
</tr>
<tr>
<td>FIRST</td>
</tr>
<tr>
<td>Ueqa</td>
</tr>
</tbody>
</table>

3. Data Classification and Data Security Policy

In order to sufficiently de-identify or anonymize records, the fields which will be protected must be defined and codified in an organization’s data security policy according to industry or regulatory compliance mandates. The data classification schema should specify the minimum list of fields that must be protected (such as SSN, Credit Card PAN, DOB, Last Name, etc.) and a list of recommended additional fields for less secure or more broadly accessed systems. These additional fields (such as First Name, City, Email, Phone Number, etc.) would then also be protected in situations where it is clearly warranted.

Even after de-identifying the most sensitive identifying fields, a clever data analyst could craft carefully constructed WHERE predicates and filters on the remaining unprotected fields to narrow the result set down to a single or small number of individuals, if they know enough about the target. However, this type of activity is highly abnormal and easy to detect or block with basic activity monitoring and anomaly detection.

A good data classification schema should also specify which data should be protected in transit, at rest and in use. The method of protection will more likely be a platform specific decision using a centralized policy driven approach that supports multiple data protection options.
It’s also important to consider protecting data at rest, in transit, and in use. Each of these states are present in different areas of the enterprise, and provide different challenges for performance and usability.

3.1. Protecting Data at Rest

At rest protection depends on the media and circumstance. Full media encryption (coarse-grained) is the obvious choice for portable or removal media such as off-site archive tapes or laptop hard drives or cell phone flash memory cards. At rest encryption at a fine-grained field or column level has particular disadvantages: Data type changes to VarByte (binary), larger field size, support issues for systems or applications not supporting binary data fields, etc.

Tokenization is an excellent solution for at rest data protection for many reasons. Data can readily move between systems, transparent to databases or applications where access to original cleartext values are not required. No changes are made to data type, field size or supported character sets. In addition, some applications and data repositories do not support VarByte data, so tokenization may be the only option.

3.2. Protecting Data in Transit

In transit is most often the domain of network traffic encryption protocols (SFTP, HTTPS, SSL, TLS), but field level protection such as tokenization or encryption can also be used to add an additional layer of security for data as it flows between systems.

3.3. Protecting Data in Use

The biggest challenge to protecting these particularly sensitive data fields is when they are in use by users and business processes. Typically only 1% to 3% of the total data volume in a large database warrants the application of fine-grained protection. In addition, up to 80% to 90% of all business analytics can be performed on the data in its protected form. Therefore, only 10% to 20% of the access to only 1% to 3% of the data will require any additional processing overhead.

Business users and organizations are consistently surprised at how infrequently sensitive data fields like SSN or Credit Card PAN actually need to be accessed in full cleartext. Both encryption and tokenization can be used without compromising referential integrity. A unique primary index field will still be a unique primary index. A tokenized field can still be used as an Index Column or Primary/Foreign Key Pair (PK/FK). The same database joins can be performed on protected columns without unprotecting the data.

In many ways, fine-grained security can become a business enabler by allowing legitimate business users broader access to data while simultaneously improving the overall security posture and achieving policy and regulatory compliance. The same applies to analytics performed within less mature environments, such as Hadoop, or data stored and processed through Cloud Services or Software as a Service (SaaS) providers.

3.4. Consistent Data Security Policy

Another important consideration for a data-centric approach is being able to apply data security policy consistently across all data states, in all environments throughout the organization. The same protection method(s), limited access rights, accountability and tamper-
proof audit trail needs to be applied consistently, all the way upstream and downstream in the data flow. A customer SSN first entered into a web form should remain protected on the web server, the application server, enterprise data warehouse, data lake, or archive media. The data should be equally protected, consistent with policy at all times from cradle-to-grave regardless of the platform used for data processing, analytics or storage.

4. Deciding Which Method is Most Appropriate

In many circumstances, the choice of protection method is obvious. For instance, using data obfuscation when copying data tables containing sensitive data from a secure production environment to a less secure development environment, or, using masking to hide the characters being typed into a web form field containing a password. Situations where data masking or data obfuscation are the obvious choice are normally straight forward and are not the focus of this document.

The choice of data-centric data protection becomes much more challenging when more complex environments with multiple conflicting variables are involved. The most challenging decision typically comes down to when to tokenize and when to encrypt. This is not a black or white decision. It is better to visualize a continuum where tokenization and encryption are on opposite ends each representing the best option where the other would be impractical or a poor fit, and a grey area between where either protection method could be used.

An enterprise solution must support multiple options to provide the necessary flexibility to protect all sensitive data and meet or exceed all unique data privacy, protection, and governance requirements.

4.1. Internal Policy or Regulatory Compliance Requirements

In some situations internal policy or regulatory security standards may mandate one method over the other. Today, most regulations such as PCI DSS and HIPAA allow both methods but these standards usually lag behind newly available or emerging data protection technology.

4.2. Fields with Embedded Logic

If there is logic embedded in specific character positions within a field, e.g. Credit Card PAN or SSN, that need to be accessed frequently, then tokenization is probably a better fit. For example, exposing the first two and last four digits of a credit card PAN or the last 4 of a SSN while protecting the rest of the field enables the vast majority of access to the column to only need access to the protected form. Tokenization provides a lot more flexibility in configuring rules specific to different data field contents.
4.3. Structured vs. Unstructured Data

The more standardized and consistent the contents of a field, the better the fit for tokenization. For example, SSN is a great candidate. Fields with data quality issues or complex, irregular fields, such as Passport Number, which could contain alpha and/or numeric characters in different positions, in different lengths are more complicated to configure for tokenization.

Tokenization is also more sensitive to data quality issues. If a field is supposed to contain only numeric digits and records are added that contain alpha characters for example, the token lookup tables would have to support an alphanumeric character set versus a much smaller numeric digits only character domain. However, tokenization can easily be configured to bypass or skip over specific characters, such as the dashes in a SSN field.

4.4. Field Size

When protecting a particularly small field (1 or 2 characters) encryption is the only option. Even T/F logical fields can be encrypted (with an initialization vector). Tokenization is limited by the width of the token lookup table used and is generally only practical for fields of three or more characters. Very large fields like freeform comment fields (CLOB) greater than 100 characters that could contain hundreds or thousands of characters are also not good candidates for tokenization.

If a protected field must be accessed most frequently in cleartext, encryption can also be faster than tokenization, especially for large fields, as encryption requires the same processing for a 2 character field as for a 16 character field. However, for standard, structured fields of limited length (3 to 15 characters), tokenization is more often the ideal solution.

4.5. Data Residency

Data residency regulations are becoming a significant challenge for data that must cross borders, as in some cloud applications. Often they require data to be de-identified, but in some cases, they do not allow particular sensitive data to cross borders at all in the original clear text form, or even encrypted. In such cases, only tokenization can be used, as the data is considered replaced, rather than encoded.

4.6. Cloud Hosting

The growing use of cloud services, Application Service Providers (ASP), Software as a Service (SaaS) and other outsourcing of data center operations, database or application hosting introduces another significant dimension to be considered and managed. Cloud services can be provided in a private cloud or public cloud. Data may also reside on shared or dedicated infrastructure and servers. This ultimately results in a shift in data security management from managing technical controls to managing contracts that stipulate the implementation of equivalent technical controls. CISO’s need to shift from managing security technology to incorporating their industry and policy compliance requirements into cloud hosting contracts.

There are several options to consider for protecting sensitive or regulated data while leveraging external cloud services besides the obvious shift to more of a contractual management approach. This can result in security being a business enabler if done right. New, lower cost data processing opportunities may be enabled by the CISO finding a way to achieve the same or improved data protection and compliance through a cloud services model.
Remember the old adage, “Outsource everything but common sense and security.” For heavily regulated industries such as finance and healthcare, outsourcing security may be the only choice, but that doesn’t mean you should abandon common sense measures.

### 4.7. Hadoop & Big Data

Hadoop and associated technology offers great promise for organizations to reduce data analytics costs and achieve scale when leveraging big data. However, this is also a very new technology where the focus has traditionally been on scalability and performance, with data security or privacy as only very minor considerations.

Although the emphasis is changing in the open source community and the leading Hadoop distribution vendors are increasingly focusing on data security features and functionality, Hadoop still has a long way to go to match the security features and functionality provided by traditional RDBMS data repositories. Adding an additional data-centric layer of security becomes even more important when there are fewer traditional layers of security to rely on.

### 5. Data-Centric Security Solution Checklist

Each of the following are important functional and operational requirements to look for in a data-centric, fine-grained data protection solution:

- A single, centralized solution that works across all core platforms
- Scalable, flexible protection methods
- Separation of duties between DBA/Sys Admin and Security Admin
- Enterprise-grade encryption key or token lookup table management
- Tamper-proof audit trail
- Transparent (as possible) to authorized end-users
- High availability (HA)
- Accountability at the end-user level
- Optional localized or remote unprotect functionality

### 6. Summary

No single data protection method is ideal for all situations. It is important to select an enterprise solution that provides comprehensive functionality, a flexible range of data protection options, broad platform and data type support, and with a proven record of success in production implementations.
Also keep in mind that one method by itself may not address all of your needs, and that a combination of multiple methods may better address a particular use case. For instance, you may choose a non-data-centric method such as volume encryption for data at rest, while protecting certain structured sensitive data fields with tokenization for security in use.

Due to the variety of data being generated today, in addition to the increased adoption of new platforms, flexibility may be the most important aspect to your data security solution. Careful review of your requirements should allow you to easily match them with the appropriate data security methods, and ensure that your selected security vendor or in-house solution includes everything you need to meet the requirements.

7. About Protegrity: Proven Experts in Data Security

Protegrity is the only enterprise data security software platform that leverages scalable, data-centric encryption, tokenization and masking to help businesses secure sensitive information while maintaining data usability. Built for complex, heterogeneous business environments, the Protegrity Data Security Platform provides unprecedented levels of data security certified across applications, data warehouses, mainframes, big data, and cloud environments. Companies trust Protegrity to help them manage risk, achieve compliance, enable business analytics, and confidently adopt new platforms.

Protegrity is headquartered in Stamford, Connecticut USA, with regional offices around the world.

For additional information visit www.protegrity.com or call 1.203.326.7200.

Copyright © 2015 Protegrity Corporation. All rights reserved. Protegrity® and the Protegrity logo, are trademarks of Protegrity Corporation. All other trademarks are property of their respective owners.