UK export Licence: OGE2020/000408

zTeros Light Use Case

19-Jun-20

UK export Licence: OGE2020/000408

Table of Contents.

Contents

Table of Tables.

Table of Figures.

UK export Licence: OGE2020/000408

INTRODUCTION

Company

GBR14 Ltd is a UK based company that provides innovative data confidentiality solutions for cloud computing systems and application.

zTeros Light Product

The product discussed in this use case application note is zTeros light.

zTeros light provides a simple single instance solution for data at rest confidentiality and security for individual records on a cloud system. zTeros Light is differs in in approach to data at rest, in that instead of using a single key to encrypt a whole volume, zTeros Light encrypts every record or file each with its own key. What this does is add defence in depth to live operation. Crypt controlled file or record access reduces the impact of data aggregation to single records and provides an opportunity for new application to use keys combined with permissions to enforce access control with strong security mechanisms.

zTeros light is designed to be simple to use, and with only 4 commands within the API, zTeros Light is probably the simplest data confidentiality product ever launched in the cloud computing market. Applications which are, or include, Archiving, Data Storage, Data-migration, Database and AWS data storage solutions, can benefit from the zTeros Light encryption service.

Key points:

- zTeros Light is available on the AWS Marketplace.
- zTeros Light provides an Information Security Confidentiality service for data-oriented web and cloud-based applications.
- zTeros light simplifies the security configurations for data storage, whilst adding flexibility and privacy for the users.
- zTeros Light provides data at rest confidentiality only.
- zTeros Light supports HTTPS connectivity, for traffic confidentiality.
- zTeros Light runs on a single EC2 instance and can be scaled.

zTeros Light encrypts each data object presented to it using a Key Variable for each data object. Then stores the encrypted data object using an anonymised index in a MongoDb instance. To put it another way, if you have a million data objects stored using zTeros Light, it means that a million keys have been used, one for each object.
After encryption, the indexed storage is the performed by a MongoDb Secure instance under the users control, providing full management and back up of the store as if it were a normal database management task.

The current version's encrypted data interface connects "as a DB records system" to MongoDB. An upgrade will be available shortly which additionally interfaces "as a file system" to AWS Elastic Block Store (EBS)and AWS S3 Bucket; and "as a DB record system" to AWS DynamoDB.

Conditions of use.

Our export licence OGE2020/000408 only covers UK, USA, CAN, AUS and NZ. If you need to install this application on an AWS zone outside these areas, please contact us.

UK export Licence: OGE2020/000408

ZTeros Light Restful Command set:

Command	Structure	Description
POST	http(s)://your-url/EncryptionService/records/ Parameters: None Body: None	Creates a new record, returns the RecordUID of the new (public) Example response record {"RecordUID":"12345678abcde"}
GET	http(s)://your-url/EncryptionService/records/{RecordUID} Parameters: None Body: None	Retrieves record {UID} Example response { "Completed": "true", "RecordUID": "12345678abcde", "Data": "My Data"}
PUT	http(s)://your-url/EncryptionService/records/{RecordUID} Parameters: None Body: New Data in your preferred format.	Updates record {UID} Example response { "Completed": "true", "RecordUID": "12345678abcde"}
DELETE	http(s)://your-url/EncryptionService/records/{RecordUID} Parameters: None Body: None	Deletes record {UID} Example response {"Completed":"true", RecordUID": "12345678abcde "}

Table 1 zTeros Light REST Command Set

A typical zTeros Light deployment:

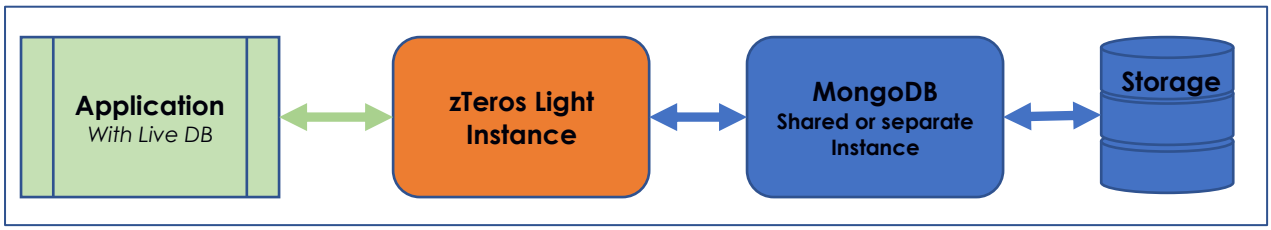

Figure 1 A typical zTeros Light deployment

Your Web application connects to the zTeros Light instance using a HTTPS connection using the 4 restful commands.
The MongoDb application is supplied installed on the zTeros light in favour of lower EC2 system costs. However better control, scaling and performance is achieved if a separate EC2 instance is used for MongoDb.
Although every data record presented to the MongoDb instance is encrypted, a HTTPS interface and a tight binding is recommended to provide additional traffic security.
The MongoDb storage is most likely Elastic Block Storage (EBS), if this is the case the free data at rest encryption service should be enabled.
Reference: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html

UK export Licence: OGE2020/000408

Using zTeros *Light*

A typical application installation is show below, where the application uses a small local scratch Db to provide referenced and fast searchable access to the live data.

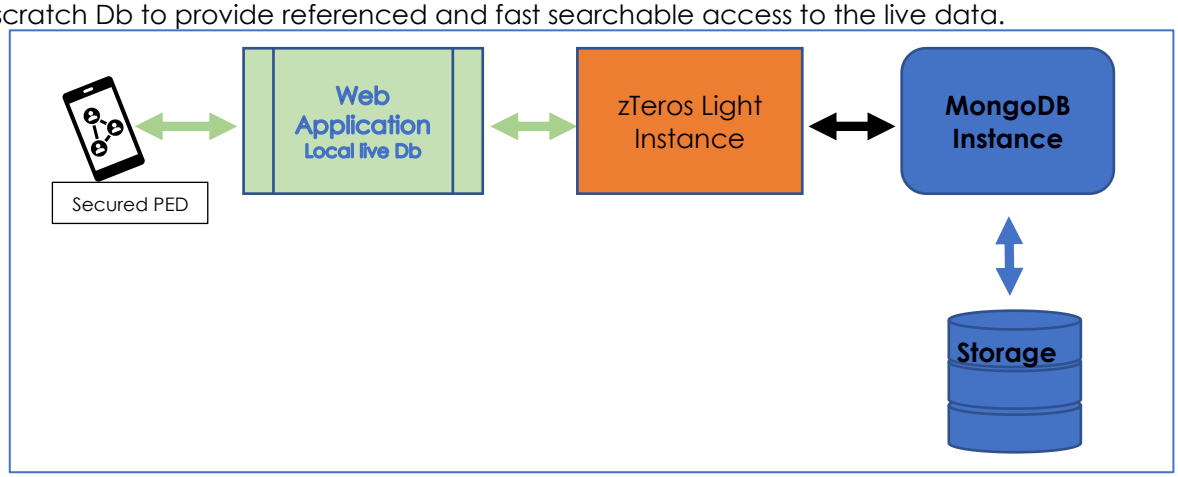

Figure 2 Typical Installation

zTeros Light has only four commands, which perform create, read, write and delete. These commands enable an application to store database records or files in individually encrypted form.
When a record is created by a database an index ID is returned to allow subsequent access to the record. This Unique IDentifier (UID) is a digitally signed 256 bit random number. It is signed to enable the rejection of fake UID's without reference to the back-end storage, saving bandwidth and reducing security risks.

Note: Signing Keys.

The signing and verification keys are generated when a new instance is created and are stored in two files in the instance's configuration files. The signing key is in an admin access protected location, the verification key is in a more accessible area. If desired, your admin can generate new signing and verification keys using the AWS Key Management Service (KMS) and overwrite the keys in the configuration files.

The UID's are the only way the application can recover any data and hence have value and their handling needs consideration; more is discussed later in this document. The advantage is that a USER can only access data that they have been explicitly be given the UID to. A Hacker or bent user CAN NOT access any other data in the database no matter how hard they try. A Hacker could of course access the Applications own local live Db, and here it is important that the data in the local live Db is strictly minimised and managed in accordance with the web applications environment.

With zTeros Light your applications live data can be scaled to mitigate the Threat model:

- If the web application is part of the office network which is protected and trusted a single application can be loaded with every office users live data.
- if the web application is loaded on the user's PED which is portable and out in a public field environment, the Local Db can be limited to only the user's data with no issues.
- If the web application services the public via their personal IT or PED, the web service could operate zero knowledge. The public customer supplies their UID which opens just the customers data and nothing else.

zTeros Light Scaling

zTeros Light is also scalable, on instance configuration you can choose to use an existing signature or create an instance with a new signature. If you reuse signatures across several zTeros light instances you can build systems which each instance is optimised to it's application and environment, and also you can load share. Yet still deploy a single store.

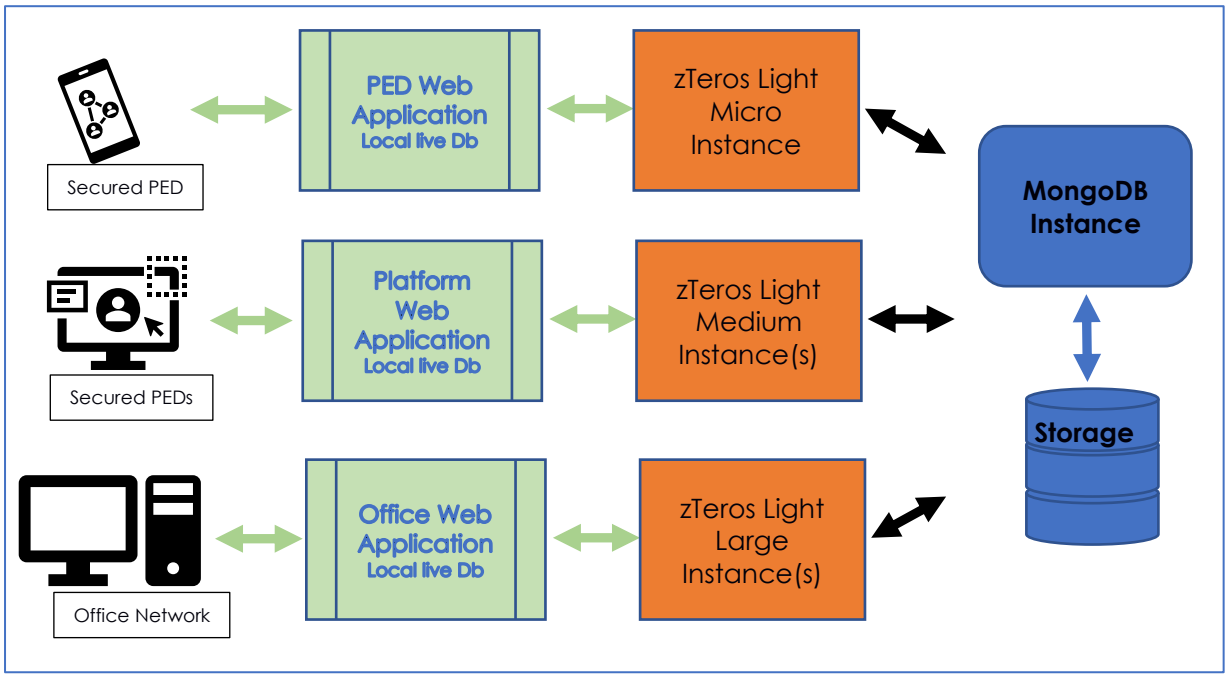

Figure 3 Typical Scaled Installation

With any scaled cloud system that uses elastic storage, eventual consistency can be an issue. zTeros is no different, what is ultimately stored is whatever is wrote last, and eventually all nodes will see that stored data.

Live work sharing, assume there are two Users teaming a live problem. To do this with zTeros the two users would access the same UID to work a technical issue.

- If the two users are on the same web application service, the shared experience will be instant.
- If the two users are connected to the common encrypted store via the two isolated web applications, then both web applications would need to perform regular refreshes against the encrypted store, and there will be refresh cycle delays.

If you mix large and small instances your application must limit the object size or otherwise chunk the object size to a common size set by the smallest instance's memory capability. See user instance documentation for details.

UK export Licence: OGE2020/000408

Using zTeros Light UIDs in a structured form.

The zTeros Light UIDs are both links your data records and the keys to your records. How you utilise them in your applications is up to your creativity.
Here are a few tips to help your creativity.

Data Isolation

An important property of zTeros Light UID's is that the data is cryptographically isolated all other encrypted data held in the store, because every data object is encrypted with its own key.
This means that when your applications permission system grants access to a UID you can be sure that the only data that can be accessed is the data contained by the UID and no other data is accessible in your data base.

In some applications your public or customer facing web service can operate zero knowledge. If your application can use something like an asset tag, ID token, or shares UID via an Email, Spread sheet, PDF or document. Your web service only needs the UID supplied by the user and then only while the session is open. In these systems you can guarantee that the user can only access the data that they have been given via the shared UID and an extremely secure public facing system can be achieved.

The isolation property also provides additional freedoms for your application. An example is the logistics and delivery last mile, where the final hands on customer or delivery may be unknown. Most current delivery systems deploy solutions which require official devices running preloaded software which means the final physical delivery service must be part of your system. In this instance if zTeros Light is used the package could carry a UID appended to a URL. The combined URL & UID would enable any mobile device to access the unique delivery information for that item and nothing else and provide the ability to receipt, redirect or return etc, providing operational freedom in the last mile.

Common Data and UIDs

Many applications have data that is common across many records, for example pictures, user guides, safety leaflets, warning labels, circuit schematics etc. With zTeros you can store each of these common objects with their own UID, and then store these UIDs in a common files object which is organised like a directory or spread sheet. This common UID would be provided to the data creation and maintenance teams in the back office. When the data creator, creates a new record for say "a jet engine", they simply copy over the required common object's UIDs into the new asset's record (UID) required fields and now any time a field user selects the Jet Engine's asset task they can access all the latest support documents for that Engine type, as well as data unique to the engine.

Prefilling Data and forms creation

You can create a UID and then prefill it with a form or set of record structures with further UIDs embedded within. Once this is done to create a new blank form, your application opens the prefilled UID, then creates a new UID and then copies the record across. The application has to repeat this for every UID held within the record too. To assist we suggest flag whether the contained UID links to a common data object as these should not have new UIDs unless you are creating a fork.

Permission chained UIDs

Your application can place permissions within your record system, this give you the

opportunity to permission guard a UID that has been placed within a record and create record access states controlled by permissions, see Figure 4 below. State chains provide an opportunity to validate the whole database security permission logic chain for each record using ether test scripts or formal proofs.

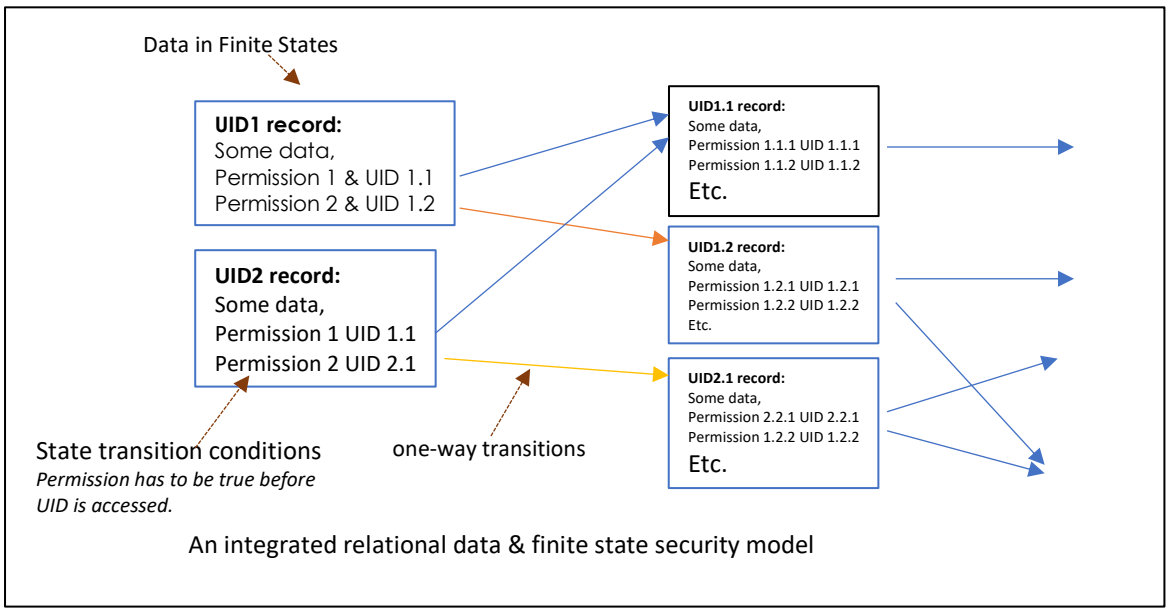

Figure 4 Using Permissions with Encrypted zTeros Light records within a database

Using the One-way connectivity to distribute anonymous data

Figure 4 also shows that the transitions between records are one way, to be in both directions a backward reference UID is required. This creates an opportunity to build applications that can share information to different communities as shown in Figure 5.

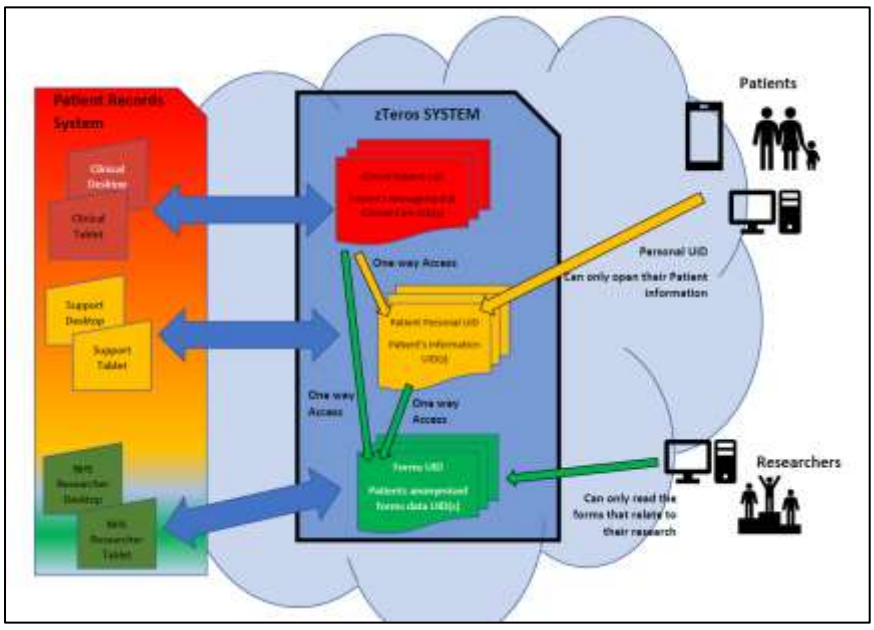

Figure 5 Using one-way connectivity

Concentration of Evil.

With encryption keys there are always traps and issues to be weary of. When you encrypt a group of keys using a Key Encryption Key (KEK), we create a Secret protecting many Secrets, and as such we have concentrated the all the evils within a single KEK. The KEK will become more valuable the more keys it is used to protect, and the evil is: the impact of the KEKs loss is increased and the information compromise becomes bigger and more costly.

zTeros Light is a single level security system, and therefore the same concentration of evils warning applies when a UID is used to encrypt a record containing a list of UIDs. We are not saying don't do it, we a saying think about it with respect to your application and respect the value of your data, try not to create a system where no application access controls have been applied to a UID which encrypts a list of all others UID's in your system.

Using Active Directory, to store zTeros Light UIDs

It is expected that inmost applications a user would need a UID which contains a list of all the UIDs they have created and or a UID which contains a list of the UIDs which have been shared to them.
AWS provides Microsoft Active Directory. Active directory provides a secure means to hold zTeros Light UID's as part of the user's login processes. There are several candidates in the existing Active Directory Schema's User Class static object attributes, or the user managed objects. If none of these suits your application, you can also create a custom attribute.

If your application requires a time limited access control UID, Active directory also supports dynamic objects. Active Directory Dynamic objects have a defined time to live feature, 1 second to a year. An application may consider the use of Active Directory dynamic object to hold a temporally shared zTeros Light UID.

Local Db storage and Data minimisation.

zTeros light only decrypts the data requested via the UID's supplied and as such live data is naturally minimised. If your application is running zero knowledge in a cloud system, or running in a small PED, the Local Scratch Db can be RAM based as the data is minimised. RAM disc file storage is supported in AWS EC2 using temporary volumes as the RAM disk based volumes. The special type of filesystem is called "tmpfs" for further information on AWS implementation see these references.

Reference: https://aws.amazon.com/about-aws/whats-new/2018/03/amazon-ecs-adds-support-for-shm-size-and-tmpfs-parameters/
Reference: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html#container_definition_linuxparameters

UK export Licence: OGE2020/000408

Compatibility and Upgrade paths.

There is an upgrade path for your encrypted data to zTeros Basic.

zTeros Basic is a true multilevel encryption system, which has added features which include cryptographic privileges for every object and permissions supported with crypt Keys. For details on zTeros basic see zTeros Basic documentation on our website. zTeros Basic will be available on the AWS marketplace in Autumn 2020 when it has completed test.

zTeros Light generated UIDs are compatible with zTeros Basic Light level UIDs.
With zTeros Basic you can upgrade your existing system and zTeros Light generated UIDs to include true multilevel crypt-controlled data.

zTeros Basic generated UIDs are compatible with zTeros Light as follows:

- zTeros Light Level data may be accessed and modified by both zTeros Light and zTeros Basic,
- zTeros Basic can promote a Light Level record to include a Multilevel Security data record.
- zTeros Basic Multilevel Security data records cannot be accessed by zTeros Light.
- zTeros Light will error "UID Locked" if any attempt is made to delete a UID which is promoted by zTeros Basic to Multilevel.