# PRE as a Service within Smart Grid City

Sbai Anass, Drocourt Cyril and Dequen Gilles
*MIS Laboratory, University of Picardie Jules Verne, Amiens, France*

Keywords: Smart-Grid, Cloud Computing, Proxy Re-encryption, Privacy, Confidentiality, Bank of Energy.

Abstract: In the context of Smart Grid Cities, legal obligations require that certain personal data must be stored in the long term and protected. To deal with confidentiality issues, we use the concept of Proxy Re-Encryption (PRE) which allows sharing encrypted data. We present the first implementation of the Chow's algorithm, and propose an optimized instantiation thanks to elliptic curves. This is the first unidirectional algorithm with CCA security that does not rely on pairing, which guarantees its high performance. This allows its use in real conditions. We have implemented it in JavaScript for direct use in a web browser by the user. In order to be able to process the data asynchronously, we then define the notion of PREaaS (Proxy Re-Encryption as a Service) that also allows use in a service-oriented context.

## 1 INTRODUCTION

Demands in terms of energy continues to grow. Nevertheless, old sources of production (as nuclear, fossil ...) are less appreciated and tend to be abandoned due to the pollution that manufacturing causes. The use of renewable energy sources has become privileged and paramount. Thus, creating a new energy market becomes a great challenge for companies. A very neat management must be adapted, starting by the smart metering systems. This involves the installation of a whole infrastructure and communication networks including smart meters. This also requires a great deal of attention at the security level, whether it is to ensure the confidentiality of consumption data, integrity to avoid fraud or, above all, to be immune from attacks aimed at destabilizing production systems such as STUXNET (Matrosov et al., 2010) which target precisely the data related to a specific type of software such as SCADA. Finally this becomes a concern of national security. Recently, (Ronen et al., 2017) exploited a vulnerability in the ZLL protocol, allowing to inset a worm spreading quickly and invisibly on the network. Such a chain reaction could rattle a nuclear power station. Other interesting facts, (Brinkhaus et al., 2011) have shown that smart meters are capable of becoming surveillance devices, as long as an attacker can access the load curves stored or transmitted by smart meters. These data, can be used to identify all the devices connected to the electrical network such as the refrigerator, microwave etc. Thus, by making predictions about the energy consumed by different devices, see if there is a correlation between predictions and actual data, we can even guess what movie was viewed.

This paper deals with the secure management of large amounts of data within the context of the VERTPOM project.[1] The latter, currently being developed, consists in creating the energy bank, which is a tool dedicated to the territory energy management. It aims to maintain an optimized balance between the available energy from production (conventional and renewable energy) with regard to usage (consumption and losses), in connection with the energy storage means (Boronat, 2017). For this purpose, the energy bank uses prediction algorithms and simulations of energy production, consumption and losses on the various distribution systems. Thus the energy networks must be more responsive, flexible, and thus foster interactions between market players. These goals are partly achieved by collecting data on networks through remotely controllable sensors and devices. As shown in Fig 1, data needed by the energy bank for its smooth operation are sent back by the smart meters via the data concentrators to the servers of the network manager. Some data managed by the energy providers (EP) and other information coming from street (light sensors etc ...) are also sent to the BE. We will ensure that all the data, once produced will be encrypted and since, only the owner could ac-

[1]This work is supported by the Future Investments Program operated by ADEME.

Anass, S., Cyril, D. and Gilles, D.
PRE as a Service within Smart Grid City.
DOI: 10.5220/0007838503940401
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 394-401
ISBN: 978-989-758-378-0

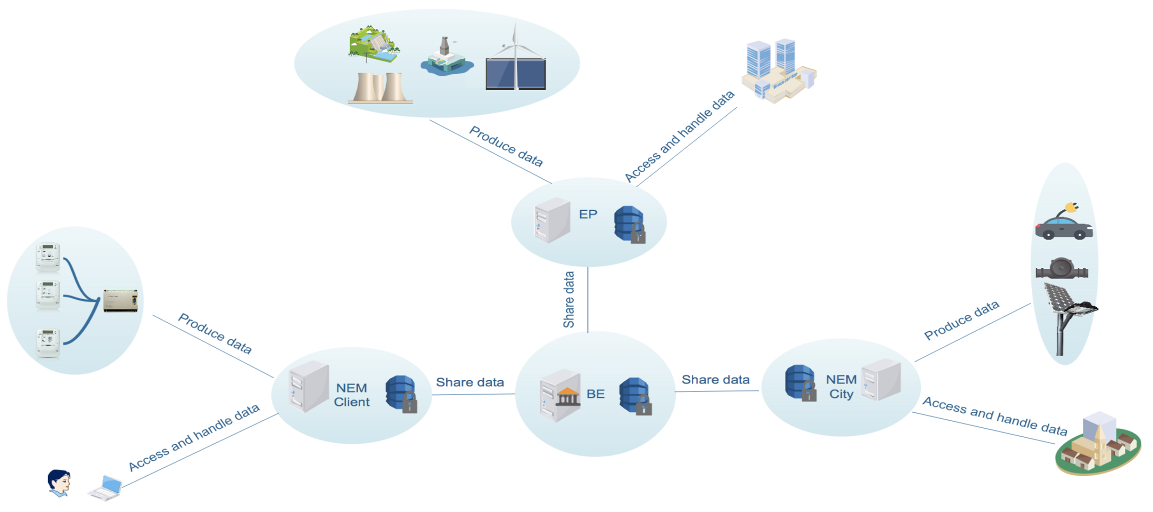

Figure 1: Architectural model for the Bank of energy.

cess the data. On the other hand, the data should be shared between different actors. Thus we propose a solution based on proxy re-encryption designed primarily to allow decryption delegation[2].

The paper is organized as follows: The next section is about the related works. We present our approach, implementation details and performances in section 3. Finally we conclude with discussion on our solution and present some perspectives.

## 2 RELATED WORKS

### 2.1 Cloud Storage

To date, cloud storage remains restrictive in terms of confidentiality. In fact, the best-known provider (e.g GoogleDrive, Dropbox, iCloud, pCloud, OVH ... ) do not ensure total confidentiality of their customers data. Either the data are encrypted under a key known by the cloud or stored in plaintext. Because these CSPs are considered as an all-trusted part. Several works exist that aim to protect privacy such as CryptDB (Popa et al., 2011) but still have some limitations. For example: it cannot be used for no-sql databases or file systems, and key management complexity increase with the number of users. ESPRESSO (Kang et al., 2014) delivers an encryption service for CSP to maintain confidentiality. It uses only symmetric encryption and requires to trust a third party or the CSPs to ensure the encryption and key management. More closer to our context

[2]Authentication issue is out of the scope of this paper

Table 1: The terms defined below will be used in the rest of the paper.

| Term | Definition |
|---|---|
| BE | Bank of Energy |
| NEM | Network Energy Manager |
| EP | Energy Provider |
| Alice, $a$ | Sender/Delegator |
| Bob, $b$ | Receiver/Delegate |
| $Pk_a$ | Public Key of a |
| $Sk_a$ | Private Key of a |
| $Rk_{a\to b}$ | Re-encryption Key from a to b |
| PRE | Proxy Re-Encryption |
| CSP | Cloud Service Provider |
| DO | Data Owner |
| DP | Data Producer |
| DC | Data Consumer |
| KEM | Key Encapsulation Mechanism |
| DEM | Data Encapsulation Mechanism |
| CCA | Chosen Ciphertext Attack |
| CPA | Chosen Plaintext Attack |
| $\xi^{asym}_{PK}$ | Asymmetric Encryption using $PK$ |
| $\xi^{sym}_{K}$ | Symmetric Encryption using $K$ |
| IBE | Identity Based Encryption |

ES4AM (Hasan and Mouftah, 2015) provides encryption schemes for smart grid AMI (Advanced Metering Infrastructure) based on symmetric encryption. In the latter case the trust remains a concern and none of these solutions allow data sharing. This issue is addressed by various solutions in the literature, starting with the broadcast encryption designed by (Fiat and Naor, 1993). Within this case, each user can access data independently from the others. This requires at the time of encryption the knowledge of who will have the privilege to access this data. Another similar approach was introduced by (Sahai and Waters, 2005) which is Attribute Based Encryption. Inspired by the work of D Boneh on the Identity Based Encryption schemes, their idea was to create a new type of IBE (Boneh and Franklin, 2001) system that they called fuzzy IBE to combine encryption and access control. In this case, access privileges are not addressed to a set of users but only to users with a specific number of attributes. Unfortunately, this does not allow for a selective sharing. As an alternative to these solutions, we choose to use the proxy re-encryption(PRE). Thanks to this scheme, we will be able to delegate decryption rights to specific entities. We found out that it was used as a cloud based solution for file sharing called Skycryptor by (Jivanyan et al., 2015). Basically, the proposed solution uses a unique symmetric key for each file to be encrypted with AES and then encrypt the key with the asymmetric public key of the user generated thanks to the PRE algorithm. The solution is a dedicated device software and is now marketed under the name of BeSafe. Each user's device has its own key pair and re-encryption is used to share files between different devices or users. But above all, the users must install the BeSafe software and use it to encrypt the data. The solution could be adapted to our problematic and used as a solution but a lot of changes need to be done. Instead, we propose the PREaaS which dosn't need any heavy client and which is more flexible, modular and transparent. We consider important to give definition and the state of art of the PRE so that the reader can better understand our choice of PRE as well as the rest of the paper.

## 2.2 Proxy Re-encryption

First appeared in 1998, it was designed by Blaze Bleum and Strauss (Blaze et al., 1998) based on the asymetric encryption scheme ElGamal. They show that it's possible to incorporate a substitution key to re-encrypt an already encrypted message without compromising it. One of the drawbacks of their method is that the system is bidirectional. That is to say, if Alice delegate decryption rights to Bob, the latter would consequently delegate decryption rights to Alice. (Ivan and Dodis, 2003) formalizes the design of proxy re-encryption schemes by categorizing these systems in two types: unidirectional and bidirectional.
Usually a PRE scheme can be defined as a tuple $\zeta : \{Setup, KeyGen, ReKeyGen, \xi^{asym}, ReEnc, Dec\}$. Where :

- $Setup(1^k) = params$ : takes as input a security parameter $k$ and generates the scheme parameters that define in general the recommended message and keys length.
- $KeyGen(params) = (Pk, Sk)$ : is the function that generate the pair public/private key.
- $RekeyGen(Sk_a, Pk_b) = Rk_{a \to b}$ in the case of unidirectional PRE, it takes as input $a$'s private key and $b$'s public key to generate the re-encryption key.
- $\xi_{PK}^{asym}(M) = C_a$ is the encryption function.
- $ReEnc(C_a, Rk_{a \to b}) = C_b$ is the re-encryption function.
- $Dec(C, Sk) = M$ is the decryption function.

In some cases we can find two more functions used for encryption and decryption in such a way that the cipher cannot be re-encrypted and only the owner of the private key can decrypt.
(Ateniese et al., 2006) gives a more formal definition for PRE and defines concretely the properties such that :

- *Unidirectional*: Delegation of decryption rights from Alice to Bob does not allow Alice to decrypt Bob's cipher.
- *Non-interactive*: The re-encryption key can be generated by Alice without interacting with Bob, and thus using only Bob's public key.
- *Transparent*: Or invisible, meaning that the delegate can not distinguish between an encrypted message and a re-encrypted message.
- *Key-optimal*: The size of Bob's secret storage must remain unchanged, no matter how many delegations he accepts.
- *Original access*: The sender can decrypt any re-encrypted message which he was originally the owner.
- *Collusion-safe*: If the proxy and Bob collude, they shouldn't get Alice's secret key.
- *Non-transitive*: The proxy can not re-delegate re-encryption rights. (e.g from $Rk_{a \to b}$ and $Rk_{b \to c}$ the proxy can not calculate $Rk_{a \to c}$)
- *Non-transferable*: The proxy and delegates can not redefine decryption rights. (e.g from $Rk_{a \to b}$ and $Pk_c$ and $Sk_b$ we can not calculate $Rk_{a \to c}$)
- *Temporary*: Bob can decipher the messages received from Alice only at a certain point in time.

## 2.3 PRE Security

Several works have been published concerning PRE security. The first functional system of an IBE-based

PRE using pairing was proposed in (Green and Ateniese, 2007) by Ateniese. These works have been implemented and patented by the authors. (Canetti and Hohenberger, 2007) proposes the first bidirectional PRE CCA Secure, where he proves the security of his scheme using the UCS method (Universally Composable Security (Canetti, 2001)) . In (Deng et al., 2008), the authors tackle the open problem presented by Canetti concerning the construction of a CCA Secure PRE without pairing. The cost of calculation decreases and their construction also makes it possible to reduce the size of the secret at the level of the re-encryption, which was not the case in their earlier scheme. (Ateniese et al., 2009) formalizes the notion of key privacy. It means that using the re-encryption key we cannot recover the identity of both the delegate and the delegator. Ateniese shows why the previous systems were not key-private and proposes a new re-encryption system considered as the first unidirectional PRE that is key private. Their construction is single-use CPA and not CCA Secure. (Chow et al., 2010) has demonstrated the possibility of conducting a CCA attack on the Shao system (Shao and Cao, 2009) and shows how to fix the issue. This leads to reduce the performance of the scheme. They then proposed their own scheme without using pairing and relying only on elGamal and the Shnorr signature. The system is unidirectional CCA Secure in the random oracle model.

## 3 OUR CONTRIBUTION

### 3.1 Architecture

If we review the global system, there are three main actors in addition to the Cloud Storage Provider (CSP[3]) as illustrated in Fig.2 (according to (Nuñez et al., 2017)):

- *Data Producers:* In our case it contains all the sensors, meters and devices generating data which has to be encrypted before storage.
- *Data Owners:* Party who is entitled to access their data and also those who can grant access to data.
- *Data Consumers:* Access with the permission of data owners to the shared information through the CSP. Mainly the BE, and other entities like the NEM.

We assume that a shared secret exists between Data Owner (DO) and Data Producer (DP) which

[3]The CSP in our use case could be a NEM client, NEM city or an EP.

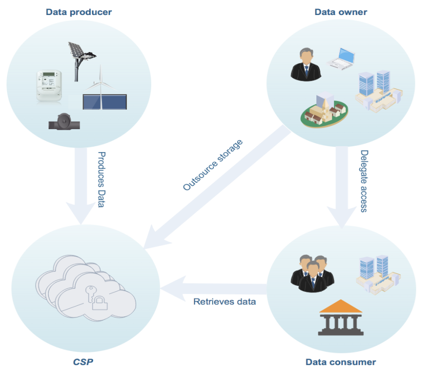

Figure 2: Main actors in a data sharing scenario.

encrypt data. Our system uses the KEM/DEM technique $(\xi^{asym}_{PK}(K)||\xi^{sym}_{K}(M))$ where the shared secret (eg. the session key '$K$') is used to encrypt data with symmetric encryption and then encapsulated within the cipher of the session key generated by the DP using asymmetric encryption of PRE. The result is sent to be stored in the cloud. When the data needs to be shared with Data Consumers (DC), the DO creates a re-encryption key and transfers it to the PRE. This one will re-encrypt them using the corresponding re-encryption keys and forward it to the recipient. Finally, the data can be consumed by DC.

#### 3.1.1 Nave Approach (Fig. 3)

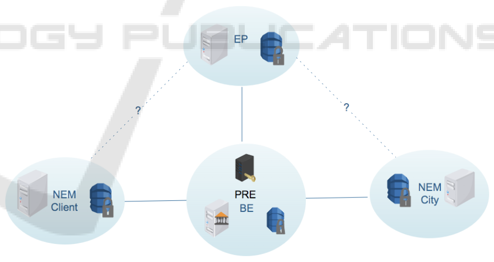

Figure 3: Proxy Re-Encryption as a Service.

Our first intuition is to put the PRE as a proxy server in the BE side. BE will send requests with its public key to the different entities passing by the PRE to retrieve the needed data. The corresponding entities should generate re-encryption keys if it belongs to owner's domain, or on the other hand should send the public key of BE to the DOs to generate them. Thus the PRE could re-encrypt the data for the BE to be treated properly, the corresponding keys (public key and its re-encryption key) will then be stored for forthcoming operations. The re-encryption procedure can be very expensive. In addition to key management issues, having only one proxy to do it can generate delays. It wouldn't be possible as well to share data between EPs and NEMs.

### 3.1.2 Second Approach (Fig. 4)

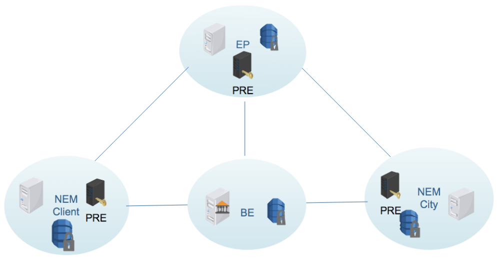

Figure 4: Second approach: PRE used by each entity.

Our second intuition is that the use of Proxy in each cloud could be more appropriate than the first approach. It allows the NEM and EP to share data with each other and relieve the bank from doing all the work. Thus, each entity will manage their own keys, re-encrypt the needed data and transfer it to the recipient as illustrated in Fig.3. Nevertheless, it still takes a long time for re-encryption due to its nature. Also, in a more general case there can be NEMs and EPs at a time. Then, it will be more complicated if every entity is forced to have their own PRE and manage a large data flow.

### 3.1.3 PRE as a Service Approach (Fig.5)

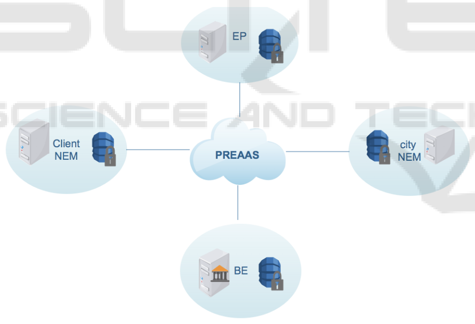

Figure 5: Proxy Re-Encryption as a Service.

Our final intuition is to propose the PRE as a service for data sharing in multi-cloud systems. Data flows already exist, and instead of NEM providing informations to the EPs, it will be enough to refer to PREaaS rather than to EPs. The potential of cloud computing would take charge of re-encryption time consumption. Our solution can be considered as a part of Security As A Service (SECaaS). Our PREaaS, has the advantage to manage only encrypted data and key management handle only public keys and Re-encryption keys. It does not affect under no circumstances the confidentiality, even if CSPs or users are corrupted and collude with the PREaaS. This is guaranteed by careful choice on the algorithms used by the service.

## 3.2 Implementation

In this section we present the algorithm used for PRE, implementation environment and the resulting performance. The algorithm used is the newest version of Chow's algorithm (Chow et al., 2010). We chose it owing to its efficiency which is due to the fact that it does not need pairings. Thanks to this survey (Qin et al., 2016), we can see the computational comparison of PRE schemes. By keeping only the schemes that are CCA secure we can conclude that Chow's algorithm is the more efficient among them. In (Selvi et al., 2017) the authors find a flaw in the security proof of Chow's construction and propose to fix it by incorporating additional information to the existing Encrypt algorithm along with ciphertext validity checks in both the Re-Encrypt and the Decrypt algorithm. Most importantly the scheme is unidirectional, collision resistant and CCA secure in the random oracle model. Two implementations were made. First we use a generic group with prime order length 3072-bit, and the second one using NIST Standard ECC p-256 (Gueron and Krasnov, 2015) thanks to SJCL (Standford Javascript Crypto Library) (Stark et al., 2013). Both correspond to the same security level that is 128-bit. We give a formal version of the Chow's algorithm with ECC below :

- $Setup(1^k)$: for 128-bit, choose a prime p of 256-bit length and an elliptic curve $\mathbb{E}_{\mathbb{F}_p}$ such that $a = p - 3$. $G = (x_G, y_G)$, a point on the curve, known as the base point with order n. The elliptic curve equation is then : $y^2 = x^3 - 3x + b \bmod (p)$. (You can find the exact values of NIST p-256 curve parameters in (Gueron and Krasnov, 2015)). Choose five hash functions: $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \mathbb{Z}^*{}_p, H_2 : \mathbb{E}_{\mathbb{F}_p} \rightarrow \{0,1\}^{l_0+l_1}, H_3 : \{0,1\}* \rightarrow \mathbb{Z}^*{}_p, H_4 : \mathbb{E}_{\mathbb{F}_p} \rightarrow \mathbb{Z}^*{}_p, H_5 : \mathbb{E}_{\mathbb{F}_p}^4 \times \{0,1\}^{l_0+l_1} \rightarrow E_{F\,p}$. The message space M is $\{0,1\}^{l_0}$, where $l_0 = l_1 = k$. $params = (p, \mathbb{E}_{\mathbb{F}_p}, G, H_1, H_2, H_3, H_4, l_0, l_1)$.
- $KeyGen(params)$:
  - Pick $x_{a,1}, x_{a,2} \overset{\$}{\leftarrow} \mathbb{Z}^*{}_p$.
  - Compute $Pk_{a,1} = x_{a,1}.G$ , $Pk_{a,2} = x_{a,2}.G$
  - Return $Sk_a$ =$(x_{a,1}, x_{a,2})$ & $Pk_a$ =$(Pk_{a,1}, Pk_{a,2})$.
- $RekeyGen(Sk_a, Pk_a, Pk_b)$:
  - Pick $h \overset{\$}{\leftarrow} \{0,1\}^{l_0}, \pi \overset{\$}{\leftarrow} \{0,1\}^{l_1}$.
  - Compute $v = H_1(h, \pi), V = v.Pk_{b,2}$ and $W = H_2(v.G) \oplus (h||\pi)$.
  - Define $rk = \dfrac{h}{x_{a,1}H_4(Pk_{a,2}) + x_{a,2}}$
  - Return $Rk_{a \rightarrow b} = (rk, V, W)$.

- $\xi^{asym}_{Pk_a}(m)$:
  - Pick $u \xleftarrow{\$} \mathbb{Z}^*{}_p, w \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Compute $D = u.[(H_4(Pk_{b,2}).Pk_{a,1}) * Pk_{a,2}]$ , $r = H_1(m,w)$ & $E = r.[(H_4(Pk_{b,2}).Pk_{a,1}) * Pk_{a,2}]$.
  - Compute $F = H_2(r.G) \oplus (m||w)$.
  - Compute $\overline{D} = u.H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)$ & $\overline{E} = r.H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)$.
  - Compute $s = u + r \times H_3(D, \overline{E}, F)$ mod($p$).
  - Return $C_a = (D, \overline{E}, F, s)$.
- $ReEnc(C_a, Pk_a, Pk_b, Rk_{a\to b})$:
  - Compute $D$ and $\overline{D}$:
    $D = s.[(H_4(Pk_{b,2}).Pk_{a,1})*Pk_{a,2}]*$
    $(H_3(E,\overline{E},F).E)^{-1}$
    $\overline{D} = (s.H_5(Pk_{a,1}, Pk_{a,2}, D, E, F))*$
    $(H_3(E,\overline{E},F).\overline{E})^{-1} =$
    $u.[H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)]$
  - Check if:
    $s.[(H_4(Pk_{b,2}).Pk_{a,1})*Pk_{a,2}]$=$D$*$(H_3(E,\overline{E},F).E)$
    (1)
    $s.[H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)]$=$\overline{D}$*$(H_3(E,\overline{E},F).E)$
    (2)
  - If the above check fail return +. Else, compute $E' = E^{rk} = (r \times h).G$
  - Return $C'_b = (E', F, V, W)$.
- $Dec(C_a, Pk_a, Sk_a)$:
  (Original CipherText)
  - Compute $D$ and $\overline{D}$. Then check if (1) & (2) holds.
  - If the condition hold, compute $(m||w) = F \oplus H_2(\frac{1}{x_{a,1}H_4(Pk_{a,2} + x_{a,2})}.E)$
  - If : $E = H_1(m,w).[(H_4(Pk_{b,2}).Pk_{a,1})*Pk_{a,2}]$ and $\overline{E} = H_1(m,w).[H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)]$ return m , else return +.

  (Transformed CipherText)
  - Compute $(h||\pi) = W \oplus H_2(\frac{1}{Sk_{a,2}}.V)$.
  - Extract $(m||w) = F \oplus H_2(\frac{1}{h}.E')$.
  - If : $V$=$H_1(h,\pi).Pk_{a,2}$ and $E'$=$(h \times H_1(m,w)).G$ return m , else return +.

We chose to use JavaScript as a core technology. And thus to be executed in the client side directly by navigator or even mobile devices without any software and also in the server side, thanks to Nodejs. For the tests we used a 2,5 GHz intel core i7, with 16 GB RAM.

Table 2. shows the time resources consumed by the different functions of Chow's algorithm for both implementations. We can see that encryption and re-encryption functions consume the most compared to keys generation and decryption. Practically, encrypt and key generation functions wont be often called. Generating re-encryption keys, depends on the number of delegations needed. But, still not constraining in terms of time consumption. On the other side, re-encryption function is called for each new user, new delegation and changed session key. Having an independent service that does the re-encryption work is lightening.

Table 2: Computational efficiency of Chow algorithm in ($ms$).

| Function | $\mathbb{F}_p$ | $ECC$ |
|---|---|---|
| KeyGen | 515 | 68 |
| ReKeyGen | 502 | 48 |
| Encrypt | 1000 | 95 |
| ReEncrypt | 967 | 89 |
| Decrypt | 979 | 78 |

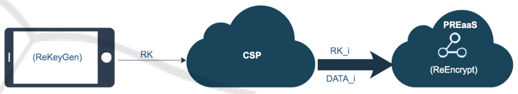

Figure 6: Tasks distribution between different environments.

An API will be developed which will act as the gateway to our service functionalities. This API does not manage any sensible data. The only concern is to prevent from DDos attacks, then a simple API key mechanism can be used as API authentication to identify the API user and the API consumers (Tang et al., 2015).

## 3.3 Data Flow

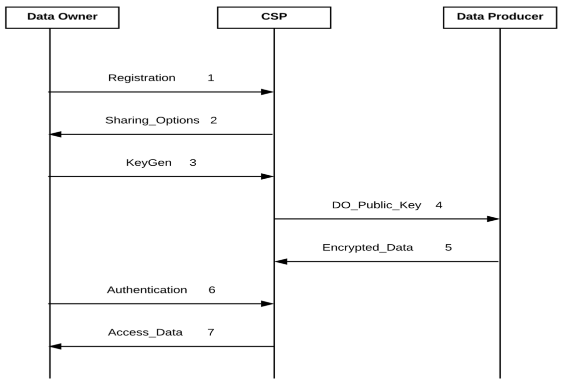

Figure 7: Interactions between DO, CSP and DP.

Figure 7 is about, in the first part the initialization process between a DO, DP and a CSP. Which consists in:

- ① The registration of the data owner with the CSP.
- ② The CSP proposing the sharing option to DOs in order to use a dedicated algorithm.
- ③ Generating a pair public/private key pair by the DO, encrypting the private key with a symmetric encryption scheme using a key derived from a password (the use of *SCRYPT* for the purpose is recommended) and sending the public key and the encrypted private key to the CSP.
- ④ The CSP storing the encrypted private key and transmitin the public key.
- ⑤ Encrypting the data with a session key that will be encrypted with the corresponding public key.
- ⑥ Authenticating DO to the CSP which will grant access in response to the corresponding data.
- ⑦ The decryption of the session key that will be deciphered by the DO with his private key.

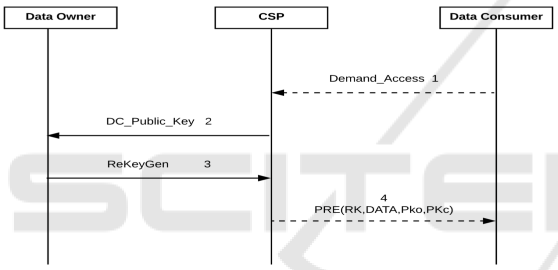

Figure 8: Interactions between DO, CSP and DC.

Figure 8, describes the sharing process between the DO and DC. The dotted arrows mean that the requests are transmitted indirectly passing by the PREaaS. The latter will at first demand access to the needed data. The CSP will forward the demand to the DO with the DC public key. As a response he creates a re-encryption Key and transmits it to the CSP. This one calls the PREaaS to re-encrypt the corresponding cipher of the session key and send the needed encrypted data.

As an illustration we provide a concrete example of data flow in Fig. 9. It includes one NEM, one client, the BE and the PREaaS. If we take into consideration all the process from registration until data consuming, then there will be 8 steps:

- 1: Registration of the NEM (who plays the role of the CSP) with PREaaS.
- 2: Notify the BE that a new NEM has been added.
- 3: Sending a sharing request to access the needed Data by the BE .
- 4, 5: PREaaS receive and forward the sharing request by sending the BE's public key to the NEM

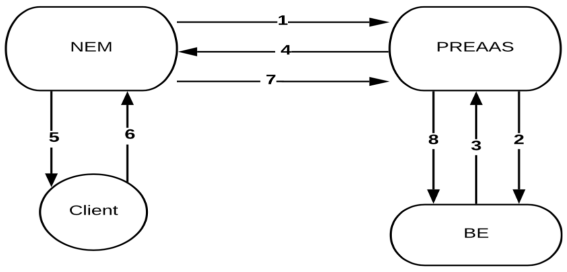

Figure 9: Data Flow.

which will send it to the client with an access request.
- 6: The client creates and transfers the re-encryption key to the NEM.
- 7: NEM authenticate with the PREaaS and return the encrypted data.
- 8: PREaaS re-encrypts the corresponding data with the key received and transfers it to the BE to be consumed.

In the case of an "automatic or predefined" sharing between the NEM and the BE, the PREaaS receives by default the new data produced from the CSP which will involve only two steps (7,8) and as soon as there is a new customer it will be asked by the NEM to create a re-encryption key that will be returned to PREaaS and added to its directory (5,6,7,8).

## 4 CONCLUSIONS

In this paper we present an original approach, which is PRE as a Service with it application to smartgrid cities. The PREaaS could be used in a more general context and applied to different scenarios. We implemented a prototype that use the Chow algorithm and evaluate the performances. We will see if it would be more appropriate to instantiate them in different manners e.g with HMAC. Which was proved to be more secure in different schemes. The PREaaS will allow the use of different PRE algorithm in future such as BBS with ephemeral keys, Ateniese scheme... And thus for scalability and flexibility purpose. As a part of the VERTPOM's project, we will work on authentication issues in multi cloud systems which we haven't treated in this work. Our future research will be focused on the security analysis of PRE, mainly on the instantiation of random oracle and its effect on the PRE schemes.

## REFERENCES

Ateniese, G., Benson, K., and Hohenberger, S. (2009). Key-private proxy re-encryption. In *Cryptographers Track at the RSA Conference*, pages 279–294. Springer.

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30.

Blaze, M., Bleumer, G., and Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–144. Springer.

Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer.

Boronat, J.-P. (2017). Vritable nergie du territoire positif et modulaire.

Brinkhaus, S., Carluccio, D., Greveler, U., Justus, B., Löhr, D., and Wegener, C. (2011). Smart hacking for privacy. In *Proceeding of the 28th Chaos Communication Congress (28C3)*.

Canetti, R. (2001). Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE.

Canetti, R. and Hohenberger, S. (2007). Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM.

Chow, S. S., Weng, J., Yang, Y., and Deng, R. H. (2010). Efficient unidirectional proxy re-encryption. In *International Conference on Cryptology in Africa*, pages 316–332. Springer.

Deng, R. H., Weng, J., Liu, S., and Chen, K. (2008). Chosen-ciphertext secure proxy re-encryption without pairings. In *International Conference on Cryptology and Network Security*, pages 1–17. Springer.

Fiat, A. and Naor, M. (1993). Broadcast encryption. In *Annual International Cryptology Conference*, pages 480–491. Springer.

Green, M. and Ateniese, G. (2007). Identity-based proxy re-encryption. In *Applied Cryptography and Network Security*, pages 288–306. Springer.

Gueron, S. and Krasnov, V. (2015). Fast prime field elliptic-curve cryptography with 256-bit primes. *Journal of Cryptographic Engineering*, 5(2):141–151.

Hasan, M. M. and Mouftah, H. T. (2015). Encryption as a service for smart grid advanced metering infrastructure. In *Computers and Communication (ISCC), 2015 IEEE Symposium on*, pages 216–221. IEEE.

Ivan, A.-A. and Dodis, Y. (2003). Proxy cryptography revisited. In *NDSS*.

Jivanyan, A., Yeghiazaryan, R., Darbinyan, A., and Manukyan, A. (2015). Secure collaboration in public cloud storages. In *CYTED-RITOS International Workshop on Groupware*, pages 190–197. Springer.

Kang, S., Veeravalli, B., and Aung, K. M. M. (2014). Espresso: An encryption as a service for cloud storage systems. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 15–28. Springer.

Matrosov, A., Rodionov, E., Harley, D., and Malcho, J. (2010). Stuxnet under the microscope. *ESET LLC (September 2010)*.

Nuñez, D., Agudo, I., and Lopez, J. (2017). Proxy re-encryption: Analysis of constructions and its application to secure access delegation. *Journal of Network and Computer Applications*, 87:193–209.

Popa, R. A., Zeldovich, N., and Balakrishnan, H. (2011). Cryptdb: A practical encrypted relational dbms.

Qin, Z., Xiong, H., Wu, S., and Batamuliza, J. (2016). A survey of proxy re-encryption for secure data sharing in cloud computing. *IEEE Transactions on Services Computing*.

Ronen, E., Shamir, A., Weingarten, A.-O., and OFlynn, C. (2017). Iot goes nuclear: Creating a zigbee chain reaction. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 195–212. IEEE.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer.

Selvi, S. S. D., Paul, A., and Pandurangan, C. (2017). A provably-secure unidirectional proxy re-encryption scheme without pairing in the random oracle model. In *International Conference on Cryptology and Network Security*, pages 459–469. Springer.

Shao, J. and Cao, Z. (2009). Cca-secure proxy re-encryption without pairings. In *International Workshop on Public Key Cryptography*, pages 357–376. Springer.

Stark, E., Hamburg, M., and Boneh, D. (2013). Stanford javascript crypto library.

Tang, L., Ouyang, L., and Tsai, W.-T. (2015). Multi-factor web api security for securing mobile cloud. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, pages 2163–2168. IEEE.