HAL open science

Attribute-Based Encryption and Its Application to a Software-Distributed Shared Memory

Oana Stan, Loïc Cudennec, Louis Syoën

To cite this version:

Oana Stan, Loïc Cudennec, Louis Syoën. Attribute-Based Encryption and Its Application to a Software-Distributed Shared Memory. 17th International Conference on Security and Cryptography, Jul 2020, Lieusaint - Paris, France. pp.625-631, 10.5220/0009890606250631 . hal-02919913

HAL Id: hal-02919913

https://hal.science/hal-02919913

Submitted on 24 Aug 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Attribute-Based Encryption and its Application to a Software-Distributed Shared Memory

Oana Stan[1], Loïc Cudennec[1,2] and Louis Syoën[1]
[1]*CEA, LIST, PC 172, 91191 Gif-sur-Yvette, France*
[2]*DGA MI, Department of Artificial Intelligence, BP 7, 35998 Rennes Armées, France*
{*oana.stan, louis.syoen*}*@cea.fr, loic.cudennec@intradef.gouv.fr*

Keywords: Distributed Shared Memory, Attribute-Based Encryption

Abstract: Due to the widespread of cloud computing, there is an increasing interest in distributed systems as well as in data privacy and security. However, unlike the multi-core processors and NUMA architectures, the Distributed Shared Memory (DSM) systems do not benefit from the numerous works on security and privacy. We argue here that their potential deployment onto distributed heterogeneous systems requires new approaches to securely manage access to shared data. In this paper we propose to rely on the Attribute-Based Encryption (ABE) techniques to tackle the problem of data security within the Software DSM. Moreover, this approach allowing to directly store encrypted data within the DSM and to manage the access control to these data is made transparent for the user. The implementation over an in-house S-DSM using libbswabe, an ABE library, shows that, as expected, the overhead is significant, but that it can also be adapted to the application requirements as a trade-off between security and performance.

1 Introduction

The recent massive adoption of cloud and edge computing increased the interest in the conception of highly-performing and in the same time safe distributed systems. In distributed architectures, distributed shared memory systems (DSM) are used to build an abstraction layer over remote memories. These systems allow to write programs that rely on the convenient shared memory programming model in which processes can concurrently access a shared space. The management of communications, the data localization and transfer are made transparent for the developer. Software-Distributed Shared Memory (S-DSM) systems are usually designed for HPC (High Performance Computing) machines such as supercomputers and clusters. HPC systems are deployed within a protected, off-line environment, the physical access being restricted to a very small list of platform administrators and the remote access being opened to a documented list of users. In these machines, applications and services are deployed under the supervision of a job scheduler and resource broker that isolates the processes from the other users as in a sandbox or virtual machine. This is the reason why most S-DSM -if not all- do not address security issues, because the context of deployment is considered safe and benevolent: the only concern is performance.

With the prevalent development of high-performance embedded computing (HPeC), some systems (Ghane et al., 2020) propose to deploy S-DSM over distributed heterogeneous architectures. These novel platforms are intended to be integrated into consumer electronic devices such as autonomous vehicles and other smart devices.

In this context, security and privacy is now of major importance as the direct environment is not safe anymore and the management of applications is not under the supervision of a trusted third-party. Furthermore, other running applications might be written by unknown developers. Therefore, S-DSM systems have to provide security and privacy mechanisms to ensure that data are only shared among authorized participants. Without any mechanism of data protection, any application with its associated users having access to this shared memory can read and modify sensitive data.

In this paper, we propose an approach for a secure and transparent API for accessing the S-DSM allowing the developers to implement secure-by-design distributed applications. It relies on Attribute-Based Encryption and it allows not only to ensure data privacy but also to control the access of the multiple users to the S-DSM system.

Attribute-Based Encryption (ABE) is an emerging cryptographic paradigm, allowing a fine grained access control to the encrypted data. Unlike the traditional public key cryptosystems where the access to the encrypted data is either all or nothing (i.e,. given the secret key, one can decrypt and read the entire message) and the key management is complicated, ABE offers a versatile way to

decrypt the data according to a specified complex policy for access control. Given the characteristics of distributed shared memory systems, more specifically, multiple users with different access rights to various sensitive data and different applications, it seems that ABE is an appropriate security solution for this particular application context. A first approach is to encrypt data before writing into the S-DSM. This approach is portable and platform-agnostic. However, it is not transparent for the developer and it requires to modify the application to manage the encryption at the user level. In this paper we propose to integrate an ABE system within the S-DSM API so that the technical aspects of the security policy are hidden to the developer. We implement this approach into an in-house S-DSM and evaluate the overhead of such an overloading of the API.

2 Related work

While being extensively studied for multi-core and multi-processor with physically shared memory and dedicated hardware-based encryption, data protection for software-DSM is quite rare in the literature. In this paper (Rogers et al., 2006) from 2006, the authors provide an analysis of the requirements to protect the DSM against hardware and coherence protocol attacks. They propose an extended version of the MESI protocol with a mix of data encryption and message authentication using AES and Galois/Counter Mode. Results based on simulation show that the overhead is around 6-8% using the SPLASH-2 benchmark. However, this requires to modify the existing DSM in order to deploy the new coherence protocol, which is not feasible for hardware-DSM and difficult to implement within software-DSM systems that do not allow multiple coherence protocols. More recently, a similar work (Khan and Henchiri, 2014) has been conducted using *signcryption* techniques but there is no report on the implementation of the solution.

ABE has been studied from 2006 (Goyal et al., 2006) with examples of applications based on audit log and targeted media broadcasting. It has thereafter been used in several contexts. In the SOUP system (Koll et al., 2014), a peer-to-peer social network is proposed in which only eligible users can access data based on ABE. This is probably the closest work to our contribution, as an ABE scheme is used over a data-sharing distributed service. However, to our knowledge there is no work that implements ABE over a S-DSM system. From a theoretical point of view, there has been significant progress for ABE constructions in terms of security guarantees and assumptions, efficiency or expressing policies ((Sahai and Waters, 2005),(Bethencourt et al., 2007), (Boyen, 2013), (Gorbunov et al., 2013), etc.). In function of the way the access policy is linked to the ciphertexts, there are two main categories of ABE schemes: the Key-Policy ABE ((Goyal et al., 2006)), in which the attributes are encrypted data and the private keys are generated according to the access policy and Ciphertext-Policy ABE ((Bethencourt et al., 2007)), in which the attributes are private keys and the access policies are associated with the ciphertexts. Other criteria which distinguish the different existing ABE schemes are the policy structure they support (e.g. Boolean formula (Goyal et al., 2006), circuits (Gorbunov et al., 2013), etc.), the underlying security assumptions they rely on (e.g. Learning with Error-LWE problem, bilinear Diffie-Hellman, etc.) or the additional challenges they address (e.g. revocation (Sahai et al., 2012), multiple authorities (Chase, 2007), etc.). However, intrinsically, all ABE schemes have a security guarantee which is the collusion resistance: a coalition of users learns nothing about the plaintext message if none of them are authorized to decrypt the ciphertext.

The analysis from Section 4 goes in more details about the existing ABE schemes but let us now give more details about the particular DSM system we want to secure with ABE.

3 Distributed Shared Memory (DSM)

Distributed computing architectures such as computing grids, clusters and micro-servers escalate the problem of shared data management due to the lack of a centralize physical memory. Distributed shared memory systems (DSM) are used to federate physically distributed memories among the system into a global logical memory. With DSM the developer can allocate and access shared data from any thread/process deployed on any node, as in a regular Posix-like programming environment.

3.1 High-level description of the Software-DSM

In this work we consider a Software-DSM (S-DSM) as presented in (Cudennec, 2018). This S-DSM is organized as a super-peer system, made of a peer-to-peer network of servers and a set of clients that run the user code. Shared data are stored into atomic pieces called *chunks*. The API provides primitives to allocate and access chunks, as well as to use common distributed synchronization objects (rendez-vous, barriers, signals..). Each access to a chunk is processed according to a specific coherence model. In this work we use a regular lazy consistency model implemented by a 4-state MESI, home-based protocol (Culler et al., 1998). This S-DSM is implemented in `C` and relies on the MPI message passing framework to manage communications, distributed bootstrapping and peer overlay.

A classical user interface to interact with the S-DSM includes the following primitives:

- `Chunk_t MALLOC(int size)`: allocates memory in the S-DSM, the function returns a *chunk* with a field `data` of size `size`.
- `void READ(Chunk_t chunk)`, `void WRITE(Chunk_t chunk)` and `void READ-WRITE(Chunk_t chunk)`: asks the system to access the given *chunk* (synchronous access primitives).
- `void RELEASE(Chunk_t chunk)`: notifies the system that the *chunk* is no longer accessed. (asynchronous release primitive)

In the user code, a call to an access primitive opens a scope in which the chunk is guaranteed to be coherent regarding the chosen coherence model. This scope is closed with a call to the release primitive. Within this scope, the chunk allows to access data in clear text. In case of a write access, the modifications are committed to the S-DSM and made available for other participants.

3.2 Privacy-preserving problem of S-DSM and threat analysis

As shown in the previous section, there are many threats and vulnerabilities with the current S-DSM design. The main issue is that communications can be intercepted and replayed with or without modification of the original message (the man-in-the-middle attack). There are three main consequences in this scenario: A) the message containing plaintext data can be discovered and modified, B) an attacker can play the role of a regular client and access shared data and C) messages can be generated to disrupt the coherence protocol, get privileges on shared data or cause a faulty state. In this work we address consequences A and B. The proposed solution does not require to modify the coherence protocol as in (Rogers et al., 2006) and it is made transparent for both the application and the S-DSM.

As for many - if not all- S-DSM systems in the literature, the security policy and access rights management are not addressed at all. In this S-DSM, all participants, as defined by a MPI process, can evenly invoke S-DSM primitives to allocate and access shared data, initiate and participate to group synchronization events. Memory chunks are identified by an UID, an *unsigned long* in this implementation. With the UID it is possible to allocate, read and write the corresponding chunk without restriction, which is the starting point of this contribution.

There are various actors which can use this memory but globally they can be classified into clients and servers. The clients correspond to the applications using the memory and they can communicate only with the servers which act as proxies for the other nodes of the network. There is no existing mechanism for an access policy or to protect sensitive data. As such, any application having the UID, i.e. the chunk identifier, can access and modify its content. This is problematic in the cases of compromised applications since it constitutes a threat for the data confidentiality and integrity of other applications using the same S-DSM instance. Moreover, in the context of heterogeneous and distant systems, this threat is quite real since it is difficult to have a perfect control for all nodes hosting the applications and to control their security.

4 ABE schemes for a secure DSM

4.1 Analysis and comparison of ABE schemes

A first step of our approach to protect the S-DSM described in the previous section was to realize a comparative analysis of the current ABE schemes and identify the most fitted for our application context. Table 1 resumes some of the main existing ABE systems and their underlying security assumptions (DBDH -Decision Bilinear Diffie-Hellman, BDHE - Bilinear Diffie-Hellman Exponent, LWE- Learning With Errors, DLWE - Decisional LWE), their type (either key policy or ciphertext -Ctxt), the way the access policy is expressed (in form of a tree, circuit or LSSS - Linear Secret Sharing Scheme matrix) as well as other particular characteristics (e.g. HABE - Homomorphic ABE allowing the possibility to process directly on ABE encrypted data, etc.)

As shown also in the table, almost all constructions from the beginning are exploiting Boolean formulas and are relying on the hardness of Discrete Log problem defined on bilinear maps. As such, they are vulnerable to quantum cryptanalysis. More recently, there are proposals of ABE schemes (e.g.(Gorbunov et al., 2013)) supporting polynomial-size circuits and relying on the quantum resistant Learning with Error problem. However, even if the last schemes give more flexibility in terms of access management, their underlying policy is complicated and thus the size of the ciphertexts and the time required for encryption and especially decryption is particularly high.

Moreover, for a first evaluation and validation of our proposal, we searched for existing ABE libraries. To the best of our knowledge, there are only a few available: (Dai et al., 2017), `libbswabe` ABE library[1] and OpenABE[2]. While the first is based on ABE scheme from (Boneh et al., 2014) and thus quantum-resistant, the execution times and the implementation constraints make it difficult to use for a fast prototyping. As such, the

[1] `http://acsc.cs.utexas.edu/cpabe`

[2] `https://github.com/zeutro/openabe`

Hypothesis	Ref	Type	Access policy	Other
DBDH	(Sahai and Waters, 2005)	Key	Threshold Structure	
	(Goyal et al., 2006)	Key	Tree-based Structure	
	(Bethencourt et al., 2007)	Ctxt	Tree-based Structure	
	(Cheung and Newport, 2007)	Ctxt	AND gates	Negation
	(Goyal et al., 2008)	Ctxt	Access tree	
	(Ibraimi et al., 2009)	Ctxt	Secret sharing schemes	Operator of
	(Liang et al., 2009)	Ctxt	Tree-based Structure	One-time signature
	(Emura et al., 2010)	Ctxt	AND gates	
BDHE	(Waters, 2011)	Ctxt	LSS matrix	
LWE	(Boneh et al., 2014)	Key	Circuit	Key homomorphic
	(Gorbunov et al., 2013)	Key	Circuit	TOR framework
	(Boyen, 2013)	Key	LSS matrix	Based on Regev
DLWE	(Brakerski and Vaikuntanathan, 2016)	Key	Circuit	Based on (Boneh et al., 2014)

Table 1: Comparison of ABE schemes

preliminary study we present here is using `libbswabe` library, based on the well established and efficient Ciphertext-Policy ABE scheme of Bethancourt et al. (Bethencourt et al., 2007). Let us now present more in details this ABE scheme.

4.2 Ciphertext-Policy ABE chosen scheme

This scheme (Bethencourt et al., 2007) is the first work on Ciphertext-Policy ABE in which the private keys of the users are described by an arbitrary number of attributes (in the form of strings) and the ciphertexts are associated with the access policy.

Access structure. Mathematically, the access structure is a monotonic access tree $\mathcal{T}$, in which the non-leaf nodes are threshold gates (AND, XOR) and the leafs are attributes.

Let $\mathbb{G}_0$ a bilinear map of prime order p with g a generator of $\mathbb{G}_0$ and $e:\mathbb{G}_0\times\mathbb{G}_0\rightarrow\mathbb{G}_1$ a bilinear map. κ is the security parameter defining the size of the groups, $\Delta_{i,S}$ is the Lagrange coefficient for $i\in\mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x)=\Pi_{j\in S,j\neq i}\frac{x-j}{i-j}$. Finally, $H:\{0,1\}^*\rightarrow\mathbb{G}_0$ is a hash function to map the attributes described as strings into random group elements.

Setup(κ). It outputs the public parameters $PK=(\mathbb{G}_0,g,h=g^{\beta},e(g,g)^{\alpha})$ and a master key $MK=(\beta,g^{\alpha})$ with α,β two random exponents from $\mathbb{Z}_p$.

KeyGen(MK,S). Takes as input a set of attributes S and outputs a key that identifies with that set:

$$SK=(D=g^{(\alpha+r)/\beta},\forall j\in S:D_j=g^r\cdot H(j)^{r_j},D'_j=g^{r_j}),$$

with random $r\in\mathbb{Z}_p$ and $r_j\in\mathbb{Z}_p$ for each attribute $j\in S$.

Encrypt(PK,M,$\mathcal{T}$).

Let Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext for message M is:

$$CT=(\mathcal{T},\widetilde{C}=Me(g,g)^{\alpha s},C=h^s,\forall y\in Y:C_y=g^{q_y(0)},C'_y=H(att(y))^{q_y(0)}),$$

with q_x a polynomial for each node $x\in\mathcal{T}$ and $att(x)$, the attribute associated with the leaf node x.

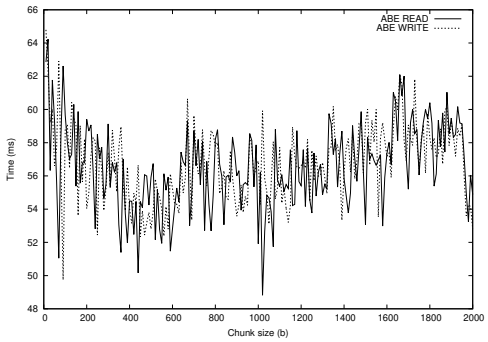
Decrypt(CT, SK). This recursive algorithm takes as input a ciphertext CT which contains the access policy, the private key SK and the public parameters PK.

For more details on the latter algorithm, as well as on the delegation capacity, we refer the reader to the original paper (Bethencourt et al., 2007).

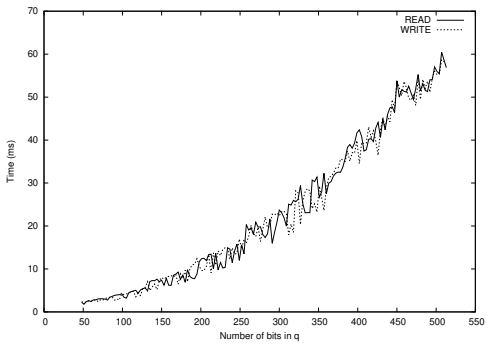
5 Generic API for a secure access to the DSM

In our prototype for a generic secure API for the S-DSM, the primitives of the ABE scheme are almost transparent for those writing applications on top of it. For encryption we use `libbswabe` library that implements the ABE ciphertext policy scheme of (Bethencourt et al., 2007). More precisely, we used a hybrid encryption of ABE with AES in CBC mode with a symmetric key of 128 bits. As such, the encryption consists in the ABE encryption of the AES key and the encryption of the data symmetrically with AES.

The access to the current S-DSM follows the entry consistency scheme. First, a call to a `malloc` function that takes the size of the data as parameter and returns a memory *chunk*. Second, a call to an access primitive, either `read`, `write` or `readwrite`, giving a chunk as parameter. This opens a scope in which it is possible to access a specific data field in the chunk. The `release` primitive is used to close the scope. Within a scope, the data field is guaranteed to be allocated and the data, in plain text, is consistent in regard of the chosen coherence



(a) Different chunk sizes ($q = 512$)



(b) Different q bit sizes

Figure 1: Access time in the S-DSM with ABE.

protocol. We overload these primitives to transparently manage ABE on each call. The `malloc` primitive requires to take into account the size overhead induced by the encrypted data. While the user still provides the data size for plain text, the overloaded primitive calculates the corresponding encrypted size to be allocated in the S-DSM. The access primitives are quite straightforward as it only needs to call the ABE encryption primitive before sending data to S-DSM servers and the decryption primitive before returning data to the user.

Preliminary experiments have been conducted on a simple producer-consumer application. A 256x256 image is written and read into/from the S-DSM between two processes. The size of a line is 256 bytes (1 byte per pixel) and the image is stored line by line into memory (a chunk per line). When setting the maximum size of chunks to a smaller value, several chained chunks are used to store the complete line. All processes have been co-located onto the same node to get rid of the network communication overhead and uncertainty.

The time to read and write the image in the S-DSM is less that 0.2ms without ABE. Figure 1a shows the time when using ABE (around 55ms) with different chunk sizes for 512 bits of security. While being a significant overhead compared to not using ABE, these values have to be placed in perspective with the communication timings when deploying onto distributed computing architectures, and most likely Ethernet and USB-based networks used by embedded devices. Furthermore, distributed runtimes are prone to hide such access costs behind pipeline parallelism.

Figure 1b presents the access time using different values for the security parameter `q`. As expected, the computing time needed to encrypt and decrypt data directly depends on this security parameter. Therefore, there is a trade-off to find between performance and security, and this trade-off depends on the application context. For example, live broadcasting of video might require to maintain a minimal framerate and stay below a given latency. The secure S-DSM can increase the security parameter up to a given point it does not break the performance requirements. Conversely, in a distributed database, the secure S-DSM can decrease the security parameter to accelerate queries down to a given value it does not violate the security policy.

6 Conclusion

Software-Distributed Shared Memory adoption in distributed heterogeneous platforms and embedded systems requires strong guarantees on data protection. In this position paper, we propose to use the Attribute-Based Encryption paradigm to manage fine grained access control to the shared data. This ABE scheme is implemented as a transparent layer standing between the application and the S-DSM programming interface. Our preliminary results on a simple reading-writing image application demonstrate the feasibility of the approach with an expected overhead (a factor scale of approximately 100) induced by the ABE cryptosystem. Several perspectives follow. The first one consists in improving the current API by using more versatile ABE schemes (e.g. with multi-level access control, allowing revocation, etc.) as well as optimizing the code to obtain better performances. Secondly, we intent to investigate the possibility to use cryptography accelerators in an opportunistic way on the heterogeneous platform. Finally, another research perspective is the exploration of an automatic tuning of the security parameter in our API for the S-DSM to fulfil the security and performance requirements, while reducing the energy consumption. As such, in function of the application context one could choose to target a lower security level but higher performances (e.g. a real-time video broadcast application) or, on the contrary, to guarantee a higher security while accepting a degradation in the overall computation (e.g. a data storage application).

REFERENCES

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, SP '07, pages 321–334, Washington, DC, USA. IEEE Computer Society.

Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., and Vinayagamurthy, D. (2014). Fully key-homomorphic encryption, arithmetic circuit abe, and compact garbled circuits. `http://eprint.iacr.org/2014/356`.

Boyen, X. (2013). *Attribute-Based Functional Encryption on Lattices*, pages 122–142. Springer Berlin Heidelberg, Berlin, Heidelberg.

Brakerski, Z. and Vaikuntanathan, V. (2016). *Circuit-ABE from LWE: Unbounded Attributes and Semi-adaptive Security*, pages 363–384. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chase, M. (2007). Multi-authority attribute based encryption. In *Theory of Cryptography*, pages 515–534. Springer Berlin Heidelberg.

Cheung, L. and Newport, C. (2007). Provably secure ciphertext policy abe. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 456–465, New York, NY, USA. ACM.

Cudennec, L. (2018). Software-distributed shared memory over heterogeneous micro-server architecture. In *Euro-Par 2017: Parallel Processing Workshops*, pages 366–377. Springer International Publishing.

Culler, D., Singh, J., and Gupta, A. (1998). *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1st edition. The Morgan Kaufmann Series in Computer Architecture and Design.

Dai, W., Doröz, Y., Polyakov, Y., Rohloff, K., Sajjadpour, H., Savaş, E., and Sunar, B. (2017). Implementation and evaluation of a lattice-based key-policy abe scheme. Cryptology ePrint Archive, Report 2017/601. `http://eprint.iacr.org/2017/601`.

Emura, K., Miyaji, A., Omote, K., Nomura, A., and Soshi, M. (2010). A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. *Int. J. Appl. Cryptol.*, 2(1):46–59.

Ghane, M., Chandrasekaran, S., and Cheung, M. S. (2020). Towards a portable hierarchical view of distributed shared memory systems: Challenges and solutions. In *Proceedings of the 11th International Workshop on Programming Models and Applications for Multicores and Manycores*, PMAM'20.

Gorbunov, S., Vaikuntanathan, V., and Wee, H. (2013). Attribute-based encryption for circuits. `http://eprint.iacr.org/2013/337`.

Goyal, V., Jain, A., Pandey, O., and Sahai, A. (2008). Bounded ciphertext policy attribute based encryption. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II*, ICALP '08, pages 579–591, Berlin, Heidelberg. Springer-Verlag.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA. ACM.

Ibraimi, L., Tang, Q., Hartel, P., and Jonker, W. (2009). Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In *Proceedings of the 5th International Conference on Information Security Practice and Experience*, ISPEC '09, pages 1–12, Berlin, Heidelberg. Springer-Verlag.

Khan, M. and Henchiri, M. (2014). An effective approach of data security for distributed shared memory multiprocessors. *International Journal of Computer Science and Information Technologies*, 5:4104–4110.

Koll, D., Li, J., and Fu, X. (2014). Soup: An online social network by the people, for the people. In *Proceedings of the 15th International Middleware Conference*, Middleware'14, pages 193–204. Association for Computing Machinery.

Liang, X., Cao, Z., Lin, H., and Xing, D. (2009). Provably secure and efficient bounded ciphertext policy attribute based encryption. ASIACCS '09, pages 343–352, New York, NY, USA. ACM.

Rogers, B., Prvulovic, M., and Solihin, Y. (2006). Efficient data protection for distributed shared memory multiprocessors. In *2006 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 84–94.

Sahai, A., Seyalioglu, H., and Waters, B. (2012). Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO 2012 - Volume 7417*, page 199–217, Berlin, Heidelberg. Springer-Verlag.

Sahai, A. and Waters, B. (2005). *Fuzzy Identity-Based Encryption*, pages 457–473. Springer Berlin Heidelberg, Berlin, Heidelberg.

Waters, B. (2011). Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. PKC'11, pages 53–70, Berlin, Heidelberg. Springer-Verlag.