Innovative Science and Technology Publications

International Journal of Future Innovative Science and Technology
ISSN: 2454- 194X Volume - 2, Issue - 2

Manuscript Title

Uniqueness -Based Circulated Verifiable Data Control in Multi-Cloud Storage

S.Saranya[1], M.Balakrishnan[2]

[1]P.G Scholar, [2]Associate Professor
[(1&2)]Selvam College of Technology, Namakkal.

May – 2016

www.istpublications.com

International Journal of Future Innovative Science and Technology, ISSN: 2454- 194X
Volume-2, Issue-2, May - 2016 editor@istpublications.com

Uniqueness -Based Circulated Verifiable Data Control in Multi-Cloud Storage

S.Saranya[1], M.Balakrishnan[2]
[1]P.G Scholar, [2]Associate Professor
[(1&2)]Selvam College of Technology, Namakkal.

ABSTRACT
Remote data integrity checking is of crucial impor-tance in cloud storage. It can make the clients verify whether their outsourced data is kept intact without downloading the whole data. In some application scenarios, the clients have to store their data on multi-cloud servers. At the same time, the integrity checking protocol must be efficient in order to sav e the verifier's cost. From the two points, we propose a novel remote data integrity checking model: ID-DPDP (identity-based distributed provable data possession) in multi-cloud storage. The formal system model and security model are given. Based on the bilinear pairings, a concrete ID-DPDP protocol is designed. The proposed ID-DPDP protocol is provably secure under the hardness assumption of the standard CDH (computational Diffie-Hellman) problem. In addition to the structural advantage of elimination of certificate management, our ID-DPDP protoco l is also efficient and flexible. Based on the client's authorizat ion, the proposed ID-DPDP protocol can realize private verificat ion, delegated verification and public verification.

Index Terms—*Cloud computing, Provable data possession, Identity-based cryptography, Distributed computing, Bilinear pairings.*

1. INTRODUCTION

Over the last years, cloud computing has become an impor-tant theme in the computer field. Essentially, it takes the in for-mation processing as a service, such as storage, computing. It relieves of the burden for storage management, universal data access with independent geographical locations. At the same time, it avoids of capital expenditure on hardware, software, and personnel maintenances, etc. Thus, cloud computing at-tracts more intention from the enterprise.

The foundations of cloud computing lie in the outsourcing of computing tasks to the third party. It entails the security risks in terms of confidentiality, integrity and availabili ty of data and service. The issue to convince the cloud clients that their data are kept intact is especially vital since the clients do not store these data locally.

Remote data integrity checking is a primitive to address this issue. For the general case, when the client stores his data on multi-cloud servers, the distributed storage and integrity checking are indispensable. On the other hand, the integrity checking protocol must be efficient in order to make it suitable for capacity-limited e nd devices. Thus, based on distributed computation, we will study distributed remote data integrity checking model and present the corresponding concrete protocol in multi-cloud storage.

A. Motivation

We consider an ocean information service corporation Cor in the cloud computing environment. Cor can provide the fol-lowing services: ocean measurement data, ocean environment monitoring data, hydrological data, marine biological data, GIS

information, etc. Besides of the above services, Cor has also some private information and some public information, such as the corporation's advertisement. Cor will store these different ocean data on multiple cloud servers. Different cloud service providers have different reputation and charging stan-dard. Of course, these cloud service providers need different charges according to the different security-levels. Usually, more secure and more expensive. Thus, Cor will select different cloud service providers to store its different data. For some sensitive ocean data, it will copy these data many times and store these copies on different cloud servers. For the private data, it will store them on the private cloud server. For the public advertisement data, it will store them on the cheap public cloud server. At last, Cor stores its whole data on the different cloud servers according to their importance and sensitivity. Of course, the storage selection will take account into the Cor's profits and losses. Thus, the distributed cloud storage is indispensable. In multi-cloud environment, distributed provable data possession is an important element to secure the remote data.

In PKI (public key infrastructure), provable data posses-sion protocol needs public key certificate distribution and management. It will incur considerable overheads since the verifier will check the certificate when it checks the remote data integrity. In addition to the heavy certificate verifica tion, the system also suffers from the other complicated certifi-cates management such as certificates generation, delivery , revocation, renewals, etc. In cloud computing, most verifie rs only have low computation capacity. Identity-based public key cryptography can eliminate the complicated certificate management. In order to increase the efficiency, identity-b ased provable data possession is more attractive. Thus, it will be very meaningful to study the ID-DPDP.

B. Related work

In cloud computing, remote data integrity checking is an im-portant security problem. The clients' massive data is outside his control. The malicious cloud server may corrupt the clients' data in order to gain more benefits. Many researchers propose d the corresponding system model and security model. In 2007, provable data possession (PDP) paradigm was proposed by Ateniese *et al.* [1]. In the PDP model, the verifier can check remote data integrity with a high probability. Based on the RSA, they designed two provably secure PDP schemes. After that, Ateniese *et al.* proposed dynamic PDP model and con-crete scheme [2] although it does not support insert operation. In order to support the insert operation, in 2009, Erway *et al.* proposed a full-dynamic PDP scheme based on the authenti-cated flip table [3]. The similar work has also been done by F. Seb´ *et al.* [4]. PDP allows a verifier to verify the remote data

integrity without retrieving or downloading the whole data. It is a probabilistic proof of possession by sampling random set of blocks from the server, which drastically reduces I/O costs. The verifier only maintains small metadata to perform the integrity checking. PDP is an interesting remote data integrity checking model. In 2012, Wang proposed the security model and concrete scheme of proxy PDP in public clouds [5]. At the same time, Zhu *et al.* proposed the cooperative PDP in the multi-cloud storage [6].

Following Ateniese *et al.*'s pioneering work, many remote data integrity checking models and protocols have been pro-posed [7], [8], [9], [10], [11], [12]. In 2008, Shacham presented the first proof of retrievability (POR) scheme wit h provable security [13].

In POR, the verifier can check the remote data integrity and retrieve the remote data at any time. The state of the art can be found in [14], [15], [16], [17]. On some cases, the client may delegate the remote data integrity checking task to the third party. It results in the third party auditing in cloud computing [18], [19], [20], [21]. One of benefits of cloud storage is to enable universal data access w ith independent geographical locations. This implies that the end devices may be mobile and limited in computation and storage. Efficient integrity checking protocols are more suitable fo r cloud clients equipped with mobile end devices.

C. Contributions

In identity-based public key cryptography, this paper fo-cuses on distributed provable data possession in multi-cloud storage. The protocol can be made efficient by eliminating the certificate management. We propose the new remote data integrity checking model: ID-DPDP. The system model and security model are formally proposed. Then, based on the bilinear pairings, the concrete ID-DPDP protocol is designed. In the random oracle model, our ID-DPDP protocol is provably secure. On the other hand, our protocol is more flexible besid es the high efficiency. Based on the client's authorization, th e proposed ID-DPDP protocol can realize private verification , delegated verification and public verification.

D. Paper Organization

The rest of the paper is organized as follows. Section II formalizes the ID-DPDP model. Section III presents our ID-DPDP protocol with a detailed performance analysis. Sec-tion IV evaluates the security of the proposed ID-DPDP protocol. Finally, Section V concludes the paper.

II. SYSTEM MODEL AND SECURITY MODEL OF ID-DPDP

The ID-DPDP system model and security definition are presented in this section. An ID-DPDP protocol comprises four different entities which are illustrated in Figure 1. We describe them below:

Client: an entity, which has massive data to be stored on the multi-cloud for maintenance and computation, can be either individual consumer or corporation.

CS (Cloud Server): an entity, which is managed by cloud service provider, has significant storage space and computation resource to maintain the clients' data.

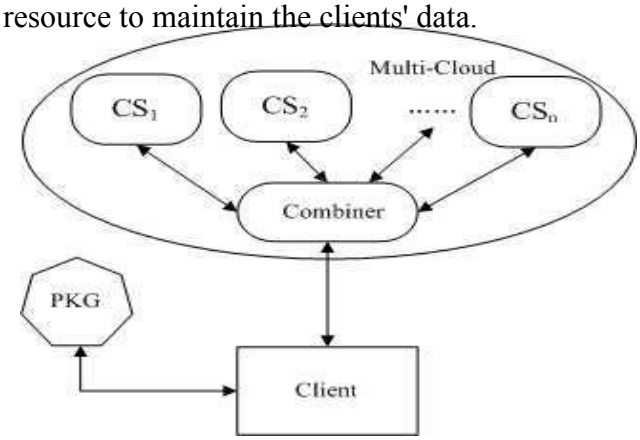

Fig. 1. The System Model of ID-DPDP

Combiner: an entity, which receives the storage request and distributes the block-tag pairs to the corresponding cloud servers. When receiving the challenge, it splits the challenge and distributes them to the different cloud servers. When receiving the responses from the cloud servers, it combines them and sends the combined response to the verifier.

PKG (Private Key Generator): an entity,

when receiving the identity, it outputs the corresponding private key. First, we give the definition of interactive proof system. It will be used in the definition of ID-DPDP. Then, the definition and security model of ID-DPDP protocol.

Completeness: for every x ∈ L, Pr[< P, V > (x) = 1] ≥ c(|x|).

Soundness: for every x 6∈L and every interactive machine B, Pr[< B, V > (x) = 1] ≤ s(|x|).

Interactive proof system is used in the definition of ID-DPDP, *i.e.*, Definition 2.

2)

Definition 2 (ID-DPDP): An ID-DPDP protocol is a col-lection of three algorithms (Setup, Extract, TagGen) and an interactive proof system (Proof). They are described in detail below.

Setup(1^k): Input the security parameter k, it outputs the system public parameters *params*, the master public key *mpk* and the master secret key *msk*.

Extract(1^k , params, mpk, msk, ID): Input the public parameters params, the master public key mpk, the master secret key msk, and the identity ID of a client, it outputs the private key sk_{ID} that corresponds to the client with the identity *ID*.

TagGen(sk_{ID} , F_i, **P**): Input the private key sk_{ID} , the block F_i and a set of *CS* **P** = **{**CS_j **}**, it outputs the tuple **{**φ_i, (F_i, T_i)**}**, where φ_i denotes the i-th record of metadata, (F_i, T_i) denotes the i-th block-tag pair. Denote all the metadata **{**φ_i**}** as φ.

Proof(**P**, C(Combiner), V (V erif ier)): is a protocol among **P**, C and V . At the end of the interactive protocol, V outputs a bit **{**0|1**}** denoting false or true.

Besides of the high efficiency based on the communication and computation overheads, a practical ID-DPDP protocol must satisfy the following security requirements:

The verifier can perform the ID-DPDP protocol without the local copy of the file(s) to be checked.

If some challenged block-tag pairs are modified or lost, the response can not pass the ID-DPDP protocol even if **P** and C collude.

To capture the above security requirements, we define the security of an ID-DPDP protocol as follows.

Definition 3 (Unforgeability): An ID-DPDP protocol is un-forgeable if for any (probabilistic polynomial) adversary **A** (malicious CS and combiner) the probability that **A** wins the ID-DPDP game on a set of file blocks is negligible. The ID-DPDP game between the adversary **A** and the challenger **C** can be described as follows:

Setup: The challenger **C** runs Setup(1^k) and gets (params, mpk, msk). It sends the public parameters and master public key (params, mpk) to **A** while it keeps confidential the master secret key msk.

First-Phase Queries: The adversary **A** adaptively makes Extract, Hash, TagGen queries to the challenger **C** as follows:

Extract queries. The adversary **A** queries the private key of the identity ID. By running

Extract(params, mpk, msk, ID), the challenger **C** gets the private key sk_{ID} and forwards it to **A**. Let S_1 denote the

extracted identity set in the first-phase.

Hash queries. The adversary **A** queries *hash* func-tion adaptively. **C** responds the *hash* values to **A**.
TagGen queries. The adversary **A** makes block-tag pair queries adaptively. For a block tag query F_i, the challenger calculates the tag T_i and sends it back to the adversary. Let (F_i , T_i) be the queried block-tag pair for index i ∈ I_1 , where I_1 is a set of indices that the corresponding block tags have been queried in the first-phase.

Challenge: **C** generates a challenge chal which defines
a ordered collection **{**ID, i_1, i_2, · · · , i_c**}**, where ID* **6∈**

S_1, **{**i_1, i_2, · · · , i_c**}** * I_1, and c is a positive integer. The adversary is required to provide the data possession proof for the blocks F_{i1} , · · · , F_{ic} .Second-Phase Queries: Similar to the First-Phase Queries. Let the Extract query identity set be S_2 and the TagGen query index set be I_2. The restriction is that **{**i_1, i_2, · · · , i_c**}** * (I_1 ∪ I_2) and ID* **6∈** (S_1 ∪ S_2). Forge: The adversary **A** responses θ for the challenge chal.We say that the adversary **A** wins the ID-DPDP game if the response θ can pass **C**'s verification.

Definition 3 states that, for the challenged blocks, the malicious *CS* or C cannot produce a proof of data possession if the blocks have been modified or deleted. On the other hand,

TABLE I -
NOTATIONS AND DESCRIPTIONS

Notations	Descriptions
G_1	Cyclic multiplicative group with order q
G_2	Cyclic multiplicative group with order q
Z_q^*	{1, 2, · · · , q − 1}
g	A generator of G_1
d	A generator of G_2
H, h, h_1	Three cryptographic hash functions
f	Pseudo-random function
π	Pseudo-random permutation
(x, Y)	Master secret/public key pair
(ID, sk_{ID})	Client's identity-private key pair
(R, σ)	Client's private key, $sk_{ID} = (R, \sigma)$
n	The block number
s	The sector number
$F = (F_1, \cdots, F_n)$	The stored file F is split into n blocks
$\tilde{F}_i = (\tilde{F}_{i1}, \cdots, \tilde{F}_{is})$	The block F_i is split into s blocks
$\tilde{F}ij$	$\tilde{F}_{ij} = h_3(F_{ij})$
CS	Cloud server
$\hat{n}$	The cloud server number
I_j	The index of the CS which stores the i-th block-tag pair
CS_{I_i}	The CS which stores the i-th block
$P = \{CS_l , 1 \le l \le \hat{n}\}$	The CS set
Tcl	The client's table which lists the storage metadata
To	The combiner's table
$u = \{u_1, \cdots, u_s\}$	The parameters picked by the client
$\phi_i = (i, u, N_i , CS_{Ii})$	The record where i denotes the i-th block, N_i denotes the block F_i 's name
(F_i , T_i)	The i-th block-tag pair
C	The combiner
V	The verifier
v_i	The permutated index $v_i = \pi_{k1}(i)$ of i
$\theta_i = (F^{\cdot(i)}, T^{(i)}_{\hat{}(i)})$, where $F^{\cdot(i)} = (F^{\cdot(i)}_1, \cdots, F^{\cdot(i)}_s)$	CS_i's response to the combiner C
$\theta = (\hat{F}, \hat{T})$, where $\hat{F} = (\hat{F}_1, \hat{F}_2, \cdots, \hat{F}_s)$	The combiner's response to the verifier V

the definition does not state clearly the status of the blocks that are not challenged. In practice, a secure ID-DPDP protocol also needs to convince the client that all of his outsourced data is kept intact with a high probability. We give the following security definition.

Definition 4 ((ρ, δ) *security):* An ID-DPDP protocol is (ρ, δ) secure if the CS corrupted ρ fraction of the whole blocks, the probability that the corrupted blocks are

detected is at least The notations throughout this paper are listed in Table I.

III. THE PROPOSED ID-DPDP PROTOCOL

In this section, we present an efficient ID-DPDP protocol. It is built from bilinear pairings which will be briefly reviewe d below.

A. Bilinear pairings

Let $\mathbf{G}_1$ and $\mathbf{G}_2$ be two cyclic multiplicative groups with the same prime order q. Let e : $\mathbf{G}_1 \times \mathbf{G}_1 \rightarrow \mathbf{G}_2$ be a bilinear map [25] which satisfies the following properties:

1) Bilinearity: $\forall g_1, g_2, g_3 \in \mathbf{G}_1$ and a, b $\in \mathbf{Z}_q$,

$$e(g_1, g_2g_3) = e(g_2g_3, g_1) = e(g_2, g_1)e(g_3, g_1)\ e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$$

2) Non-degeneracy: $\exists\, g_4, g_5 \in \mathbf{G}_1$ such that $e(g_4, g_5) = \mathbf{6}\ 1\mathbf{G}_2$.
3) Computability: $\forall g_6, g_7 \in \mathbf{G}_1$, there is an efficient algo-rithm to calculate $e(g_6, g_7)$.

Such a bilinear map e can be constructed by the modified Weil [23] or Tate pairings [24] on elliptic curves. Our ID-DPDP scheme relies on the hardness of CDH (Computational Diffie-Hellman) problem and the easiness of DDH (Decisional Diffie-Hellman) problem.

B. The Concrete ID-DPDP Protocol

This protocol comprises four procedures: Setup, Extract, TagGen, and Proof. Its architecture can be depicted in Figure 2. The figure can be described as follows: 1. In the phase Extract, PKG creates the private key for the client. 2. The client creates the block-tag pair and uploads it to combiner. The combiner distributes the block-tag pairs to the different cloud servers according to the storage metadata. 3. The verifier sends the challenge to combiner and the combiner distributes the challenge query to the corresponding cloud servers according to the storage metadata. 4. The cloud servers respond the challenge and the combiner aggregates these responses from the cloud servers. The combiner sends the aggregated response to the verifier. Finally, the verifier ch ecks whether the aggregated response is valid.

The concrete ID-DPDP construction mainly comes from the signature, provable data possession and distributed computing. The signature relates the client's identity with his private key. Distributed computing is used to store the client's data on multi-cloud servers. At the same time, distributed computing is also used to combine the multi-cloud servers' responses to respond the verifier's challenge. Based on the provable data possession protocol [13], the ID-DPDP protocol is constructed by making use of the signature and distributed computing.

Without loss of generality, let the number of stored blocks be n. For different block F_i, the corresponding tuple (N_i, CS_{li} , i) is also different. F_i denotes the i-th block. Denote N_i as the name of F_i. F_i is stored in CS_{li} where

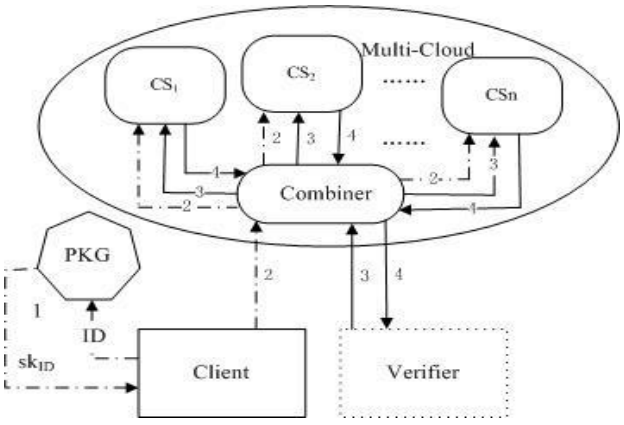

Fig. 2. Architecture of our ID-DPDP protocol

l_i is the index of the corresponding CS. (N_i, CS_{li} , i) will be used to generate the tag for the block F_i. The algorithms can be described in detail below.

Setup: Let g be a generator of the group $\mathbf{G}_1$ with the order q. Define the following cryptographic hash functions:

$$H : \{0, 1\}^* \rightarrow \mathbf{Z}_q^*$$
$$h : \{0, 1\}^* \times \mathbf{Z}_q^* \rightarrow \mathbf{G}_1$$
$$h_1 : \{0, 1\}^* \rightarrow \mathbf{Z}_q^*$$

Let f be a pseudo-random function and π be a pseudo-random permutation. They can be described in detail below:

$$f : \mathbf{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \mathbf{Z}_q^*$$
$$\pi : \mathbf{Z}_q^* \times \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\}$$

PKG picks a random number $x \in \mathbf{Z}_q^*$ and calculates $Y = g^x$. The parameters $\{\mathbf{G}_1, \mathbf{G}_2, e, q, g, Y, H, h, h_1, f, \pi\}$ are made public. PKG keeps the master secret key x confidential.

Extract: Input the identity ID, PKG picks $r \in \mathbf{Z}_q^*$ and calculates $R = g^r$, $\sigma = r + xH(ID, R) \bmod q$

PKG sends the private key $sk_{ID} = (R, \sigma)$ to the client by the secure channel. The client can verify the correctness of the received private key by checking whether the following equation holds.

$$\sigma = RY\ H(ID,R) \quad (1)$$

If the formula (1) holds, the client ID accepts the private key; otherwise, the client ID rejects it.

- TagGen(sk_{ID} , F, **P**): Split the whole file F into n blocks, *i.e.*, $F = (F_1, F_2, \cdots, F_n)$. The client prepares to store the block F_i in the cloud server CS_{li} . Then, for $1 \le i \le n$, each block F_i is split into s sectors, *i.e.*, $F_i =$

Denote $u = \{u_1, u_2, \cdots, u_s\}$. For F_i, the client performs the procedures below:

1) The client calculates F_{ij} for every sector

$$\tilde{F}_{ij} = h(F_{ij}), 1 \le j \le s.$$

2) The client calculates

$$T_i = (h(N_i, CS_{li}, i)u_j^{F_{ij}})_{j=1}^{s}$$

3) The client adds the record $\varphi_i = (i, u, N_i, CS_{li})$ to the table T_{cl}. It stores T_{cl} locally.

4) The client sends the metadata table T_{cl} to the combiner. The combiner adds the records of T_{cl} to its own metadata table T_o.

5) The client outputs T_i and stores (F_i , T_i) in CS_{li} .

- Proof (**P**, C, V):is a 5-move protocol This among and V with the input $\mathbf{P} = \{CSi\}i\in[1,n\hat{}]$, C'(*public*

After receiving the response ̂, the verifier

C. Performance analysis

First, we analyze the performance of our proposed ID-DPDP protocol from the computation and communication overhead. We compare our ID-DPDP protocol with the other up-to-date PDP protocols. On the other hand, our protocol does not suffer from resource-consuming certificate management whi ch is require by the other existing

protocols. Second, we analyze our proposed ID-DPDP protocol's properties of flexibility a nd verification. Third, we give the prototypal implementation of the proposed ID-DPDP protocol.

Computation: Suppose there are n message blocks which will be stored in nˆ cloud servers. The block's sector number is s. The challenged block number is c. We will consider the computation overhead in the different phases. On the group $\mathbf{G}_1$, bilinear pairings, exponentiation, multiplication, and the hash function h_1 (the input may be large data, such as, 1G byte) contribute most computation cost. Compared with them, the hash function h, the operations on $\mathbf{Z}_q$ and $\mathbf{G}_2$ are faster, the hash function H can be done once for all. Thus, we do not consider the hash functions h and H , the operations on $\mathbf{Z}_q$ and $\mathbf{G}_2$. On the client, the computation cost mainly comes from the procedures of T agGen and V erif ication (*i.e.*, the phase 5 in the protocol Proof(**P**, C, V)). In the phase *TagGen*, the client it makes Private verification, delegated verification and public *ver i-fication* : Our proposed ID-DPDP protocol satisfies the private verification and public verification. In the verification pro ce-dure, the metadata in the table T_{cl} and R are indispensable. Thus, it can only be verified by the client who has T_{cl} and R *i.e.*, it has the property of private verification. On some cases, the client has no ability to check its remote data integrity, for example, he takes part in the battle in the war. Thus, it will delegate the third party to perform the ID-DPDP protocol. The third party may be the third auditor or the proxy or other entities. The client will send T_{cl} and R to the consignee. The consignee can perform the ID-DPDP protocol the private key (R, σ). On the other hand, even if R is made public, the private key σ still keeps secret. The private key extraction process Extract is actually a modified ElGamal signature scheme which is existentially unforgeable. For the identity ID, the corresponding private key (R, σ) is a signature on ID. Since it is existentially unforgeable, the private key σ still keeps secret even if R is made public. Thus, the client can generate the tags for different files with the same privat e key σ even if T_{cl} and R public.

CONCLUSION

In multi-cloud storage, this paper formalizes the ID-DPDP system model and security model. At the same time, we propose the first ID-DPDP protocol which is provably secure under the assumption that the CDH problem is hard. Besides of the elimination of certificate management, our ID-DPDP protocol has also flexibility and high efficiency. At the same time, the proposed ID-DPDP protocol can realize private ver-ification, delegated verification and public verification ba sed on the client's authorization.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," ***Pervasive Computing, IEEE***, vol. 8, pp. 14–23, 2009.

[2] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in ***Proc. of EuroSys***, 2011.

[3] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel exe-cution in the cloud for mobile code coading," in ***Proc. of INFOCOM***, 2012.

[4] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: enabling remote computing among intermit-tently connected mobile devices," in ***Proc. of MobiHoc***, 2012.

[5] S. M. George, W. Zhou, H. Chenji, M. Won, Y. Lee, A. Pazar-loglou, R. Stoleru, and P. Barooah, "DistressNet: a

wire-less AdHoc and sensor network architecture for situation manaigement in disaster response," ***IEEE Communications Magazine***, vol. 48, no. 3, 2010.

[6] D. W. Coit and J. Liu, "System reliability optimization with k-out-of-n subsystems," ***International Journal of Reliability, Quality and Safety Engineering***, vol. 7, no. 2, pp. 129–142, 2000.

[7] D. S. J. D. Couto, "High-throughput routing for multi-hop wireless networks," PhD dissertation, MIT, 2004.

[8] Y. Wen, R. Wolski, and C. Krintz, "Online prediction of battery lifetime for embedded and mobile devices," in ***Power-Aware Computer Systems***. Springer Berlin Heidelberg, 2005.

[9] A. Leon-Garcia, ***Probability, Statistics, and Random Pro-cesses for Electrical Engineering***. Prentice Hall, 2008.

[10] S. Huchton, G. Xie, and R. Beverly, "Building and evaluating a k-resilient mobile distributed file system resistant to device compromise," in ***Proc. of MILCOM***, 2011.

[11] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Su, "A survey on network codes for distributed storage," ***Proc. of the IEEE***, vol. 99, no. 3, pp. 476–489, 2010.

[12] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocation for high reliability," in ***Proc. of ICC***, 2010.

[13] M. Aguilera, R. Janakiraman, and L. Xu, "Using erasure codes ly for storage in a distributed system," in ***Proc. of DSN***, 2005.

[14] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in ***Proc. of INFOCOM***, 2012.

5