Data Communication Framework for Authenticity and Integrity in IoT

Pampapathi B. M[1], Anjineyulu B[2], Tejaswini S[3], Karthik Kumar Kasalay[4], Mohammed Nayeem[5], Asst. Professor, CSE dept., RYM Engineering College, Ballari,India[1]

CSE dept., RYM Engineering College, Ballari, India [2,3,4,5]

ABSTRACT

Internet of Things has been widely applied in everyday life, ranging from transportation, healthcare, to smart homes. As most IoT devices carry constrained resources and limited storage capacity, sensing data need to be transmitted to and stored at resource rich platforms, such as a cloud. IoT applications retrieve sensing data from the cloud for analysis and decision making purposes. Ensuring the authenticity and integrity of the sensing data is essential for the correctness and safety of IoT applications. We summarize the new challenges of the IoT data communication framework with authenticity and integrity and argue that existing solutions cannot be easily adopted. We present two solutions, called Dynamic Tree Chaining (DTC) and Geometric Star Chaining (GSC) that provide authenticity, integrity, sampling uniformity, system efficiency, and application flexibility to IoT data communication. Extensive simulations and prototype emulation experiments driven by real IoT data show that the proposed system is more efficient than alternative solutions in terms of time and space.

KEYWORDS

IoT, Cloud, Authentication, Partial Data Retrieval, Sampling

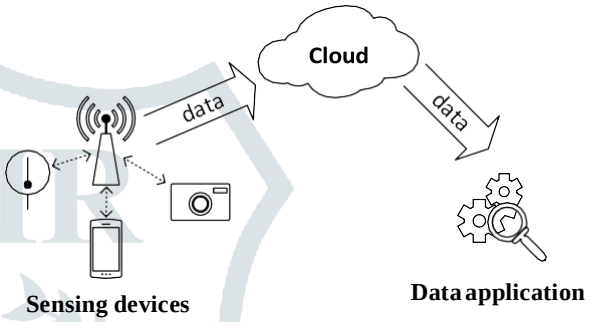

Figure 1: IoT data communication framework

1 INTRODUCTION

Internet of Things (IoT) is being widely applied in a great number of everyday applications such as healthcare [2,13], transportation [22], smart home [9, 14, 24], and surveillance systems [17,25]. Internet of Things (IoT) is fast growing at an unprecedented rate: the number of connected IoT sensing devices is expected to reach 8 billion by 2018, predicted by Cisco [29]. IoT devices generate a large amount of sensing data to reflect physical environments or conditions of objects and human beings. As most IoT devices carry constrained resource and limited storage capacity, sensing data need to be transmitted to and stored at resource-rich platforms, such as a cloud. On the other hand, analyzing historical sensing data is essential for decision making in various IoT applications [14] [32]. For example, Nest Learning Thermostat [14], a system that controls the temperature of a smart home automatically and intellectually, learns a user's preference by analyzing history data of the home. Hence IoT applications retrieve sensing data from the cloud for analysis and decision making purposes. To this end, both state of art IoT proposals [23, 24] and industrial practices [11] adopt the centralized data store residing in the cloud, as depicted in Fig. 1.

In this paper we present the design of an IoT data communication framework involving the three key entities: *sensing devices*, *cloud*, and *data applications*. We summarize the following key requirements or challenges of the IoT data communication framework, which distinguish it from traditional data collection and management methods.

Time Series Data And Event Data.
IoT sensing data can be classified into two types: time series data and event data [39]. Time series data are generated by each device for every fixed time period, such as 1 second. They are used to conduct continuous monitoring tasks such as temperature reports. Event data are generated whenever certain types of events occur, such as a vehicle appearing in a smart camera. They are used to monitor discrete events. Note event based data are more difficult to manage than time series.

1) **Data sampling.**
A common but critical problem shared by state-of-art IoT designs is that the resources for transmitting and storing data (e.g. network bandwidth, storage quota) are limited in the presence of massive IoT data

2) **Authenticity and integrity**.
Since the sensing data are stored in a third party cloud, they could be corrupted by outside attackers, malicious cloud employees [27], transmission failures, or storage loss [8]. Therefore, data authenticity and integrity, which guarantee that data are from these sensing devices and has not been modified or partially dropped, are important for trustworthy IoT applications [34].

3) **Flexible application requirements.**
Different applications may have different requirements on sensing data granularity. For example, applications like self-driving cars need fine-grained road information, while other applications like road-traffic estimation only need a few sampled data. Even if the cloud can store up to 100 records, some applications only retrieve part of them, e.g., 10 records, due to bandwidth/memory limit, or application requirements. A possible attack is that a malicious cloud operator may selectively send partial data to a user, e.g., those outlier data, and lead the user to a wrong decision. Hence we require the partial data should have verifiable authenticity, integrity, and uniformity. We call this feature as partial data retrieval.

2 PROBLEM STATEMENT

Network Model

We demonstrate the life cycle of IoT sensing data in Fig. 1. Three different kinds of entities are identified as follows.

- **Sensing devices** are distributed electronic equipments that generate IoT sensing data. They usually have limited computation, memory, and power re- sources.
- **Cloud** is an ISP or a third-party cloud provider who has rich resources and expertise in operating cloud computing services. It charges clients for data storage and data access.
- **Data applications** are software systems or devices that may request to retrieve the sensing data for analysis purposes. Different data applications may have different data requirements.

Threat Model

We assume only IoT sensing devices and data applications are trustworthy and any entities in between are subject to attack or may perform functionalities in a dishonest way.

While clients trust cloud providers to perform their services correctly, there are increasing concerns about the security of outsourced data. The security threats may be attributed to management errors or software/hardware bugs which lead to Byzantine failures. A recent report [8] describes massive cloud service outages that affect many companies and result in data corruption. Even worse, there may exist adversaries and hackers who have gained accesses to data on clouds and are able to manipulate or delete clients' data without being detected by cloud providers.

3 Dynamic Tree Chaining (DTC)

We start from the Tree chaining designed by Wong and Lam [37], one variation of Merkle tree [28]. The digest of each event report is one leaf node in the binary *authentication tree presented* in Fig. 2. The value of any internal node is computed as the hashing of the concatenation of its two children. Take the authentication tree in Fig. 2 as an example. $D12$ is the parent of $D1$ and $D2$ and $D12 = H(D1||D2)$, where $H(\cdot)$ is the message digest function, such as SHA-1
[10] or MD5 [1]. Likewise, $D14 = H(D12||D34)$ and $D18 = H(D14||D58)$. As a result, the root summarizes all the leaf nodes. The root node is regarded as the block digest and is signed by the private key to create the block signature.

The verification process is on a prevent basis. In order to verify the authenticity/integrity of an event e, the verifier requires the block signature, the position of event e in the authentication tree and the sibling nodes in the path to the root, which are all appended to event e. Basically, the verification algorithm is to replay the process to build the authentication tree and to verify the nodes in the path to the root. In this case, all the nodes in the path as well as their siblings are verified and they could be cached to accelerate the verification process. as result, the buffer space constrains the performance of DTC, which is a particularly severe problem in IoT environment where most devices possess little buffer space. More importantly, DTC provides no verifiable uniformity for event sampling and partial data retrieval, because data selection is completely executed in the cloud. If the cloud selects event samples or the partial data with bias, data applications are unaware of it.

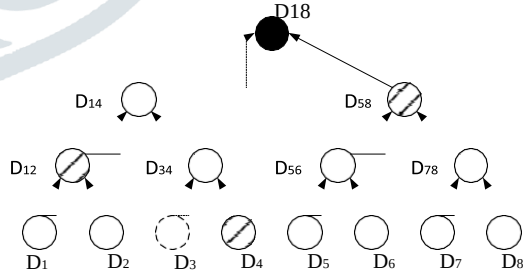

Figure 3: Visualization of numerical intervals.

4 Geometric Star Chaining (GSC)

We propose a more efficient and secure data communication framework in this paper, called Geometric Star Chaining (GSC). The basic idea of GSC is inspired by one observation that any arbitrary fraction value can be represented or closely approximated by a few number of binary digits. For instance, $5/8 = (0.101)_2$. Thus, partial data with sample rate p, where $p = 2^{-b}\boldsymbol{i}$, is equivalent to the union of multiple data blocks each corresponds to one set bit in the binary representation. One data block is called a *sample blocks* in this paper. For instance, to retrieve a sampled data from all sensing data from a device within an epoch with sampling rate 5/8, the cloud can send the data application two blocks containing

(approximately) 1/2 and 1/8 of the data respectively. Events within a same data block are either completely retrieved or not retrieved at all. Thus we can view each data block as an atomic "giant event". GSC computes one message digest for every block and concatenates these digests to a single digest for digital signature, as is depicted in Fig. 4.

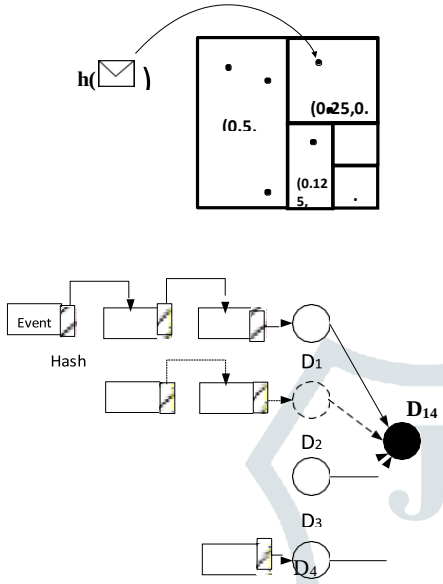

Figure 4: Illustration of GSC. Verifying the second sample block requires all events within it as well as D_1,D_2, D_3, D_4 and **signature of the root (**E_{pk}—1 [D_{14}]**)..**

Rate of each block, should also be stored and hashed with the block. In this way, the application that receives the block can verify the sampling rate.

5 Data Retrieval and Verification

A sampled fraction of sensing data is usually sufficient for most IoT applications [6]. In the network model presented in Sec. 2.1, an application requests for a certain fraction of events observed at a particular sensing device from the cloud. GSC provides verifiable authenticity, integrity, and uniformity for partial data retrieval with an arbitrary sampling rate.

6 Incorporating Budget Limit

IoT data volume is growing in an unprecedented speed over the years. With ever increasing volume of IoT data, storing all raw IoT data in the cloud poses a heavy monetary burden on the users. Since a small fraction of uniformly sampled IoT data satisfy most IoT applications, we identify the necessity to uniformly sample IoT data before storing them in the cloud.

7 SAMPLING PROTOCOL DESIGN

In this section, we describe a sampling protocol taking the budget limit into consideration, which respects the network model described in Sec.2.1. This sampling protocol introduces a new entity, called a *coordinator* (sometimes it is also called a *hub*), in the network model. One coordinator is a software working as a sampler which sits between the sensing devices and the cloud. A coordinator can be installed on an access point or a server at the edge of Internet. It maintains communications with all sensing devices on behalf of the cloud and temporarily buffers IoT data samples.

8 SECURITY GUARANTEE

We use digital signatures to verify data integration and authentication. Any inconsistency in the verification procedure indicates data in the cloud entrusted. In the sampling protocol, each sensing device maintains a counter to record the number of events that fall in a certain sample block. At the end of each epoch, the sensing device signatures both sampled events and all counters it maintains. Meanwhile, each sensing device is required to sign its counters even if it has not generated any event during an epoch. With the signature, an attacker cannot manipulate, delete or produce fake samples by modifying contents of events and counters without being detected.

9 Evaluation

We conduct extensive trace-driven simulations and prototype experiments using real dataset in this section.

Experiment Setup and Methodology

The dataset [3] used in this section includes a wide variety of data collected from the sensing devices at three real homes from May 1st, 2012 through July 31st, 2012 (the data for May 31st, 2012 and June 26th, 2012 are missing). We select sources of data comprising of time series data and event-based data generated at sensing devices: environmental information (including temperature and humidity, etc.) about homeA, homeB and homeC respectively; electrical data from dimmable and non-dimmable switches for homeA; two sets of operational data on door and furnace on/off for homeA; the data from the motion detector located at homeA. We ignore the other 5 sources of data, which are from energy meter readings because energy meters in smart grid usually utilize dedicated communication infrastructure to deliver meter readings [40]. Moreover, sizes of these data are much larger than the 7 data sources aforementioned and thus these data would overwhelm the whole system and few events from the 7 smaller-sized data sources would be sampled if the other 5 sets of data are included in the evaluation trace. We left setting the weight for each sensing device as our future work. In both the simulation and prototype emulation experiments, non-cryptographic 64-bit xxHash [16] is leveraged as the hashing function, which performs on the entire event re- port (which is one line of record). We simulate the sampling protocol driven by the 7 sets of data listed above. In the simulation, each day is one epoch and the 7 sources of data from the same day are replayed simultaneously. We vary the budget limit and evaluate its impact on the simulation results. We implement the prototypes of DTC and GSC for performance comparison. In the prototype experiment, we first test the signing and verifying performance without sampling protocol involved. We next conduct prototype experiment in a setting where sampling protocol is involved. The second prototype experiment focuses more on the potential maximal throughput of tested signature schemes, since other impact- ing factors (e.g. space) are explored in the prior prototype experiment. In the second prototype experiment, the events sent to the coordinator (which may be discarded later at the coordinator) are used for signing performance evaluation whereas the verification algorithm is fed by the events saved at the

cloud.

We utilize OpenSSL [15] to implement two widely-used asymmetric encryption algorithms, RSA [31] and DSA [4]. MD5 [1], SHA-1 or SHA-256 [10] are leveraged as the message digest function. The prototype emulation experiments are conducted on a quadcore@3.40G Linux desktop with 32GB memory and only one core is used.

Simulation Result

We first conduct one micro-scale experiment to illustrate how the sampling protocol proceeds when new events arrival, as depicted in Fig. 5. We fix the budget limit to 500 events in this micro-scale experiment. The three lines in Fig. 5 represent the number of events buffered at the coordinator, sent to the coordinator by all the 7 sensing devices and monitored at all sensing devices, respectively. The three lines vary against time in one day (May 1st, 2012). Initially, the number of events is the same for the three lines until the number of buffered events at the coordinator reaches the budget limit. At this time, approximately half buffered events are discarded, illustrated as the first vertical drop in Fig. 5. The events at coordinator then are accumulated over time until the next sharp decrease. This process repeats down to the end of this epoch. It is evident that the space used at the coordinator never exceeds the budget limit. It is worthy mention that the total number of events sent to the coordinator grows

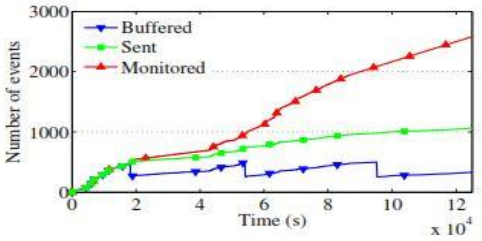

Figure 5: One-day micro-scale experiment unveiling sampling protocol.

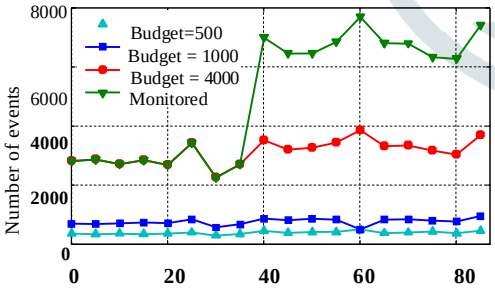

Figure 6: Number of events saved in the cloud.

Slower as time proceeds, which is a desirable property since the communication cost stays low even if much more events are monitored. Moreover, it seems that the number of events sent to the coordinator is more related to the budget than the total number of monitored events. The formal proof is left as future work. Lastly, it can be inferred from Fig. 5 is that the event occurrence rate is not constant; otherwise, more trivial solutions such as fixing sampling rate at each device are enough for the problem we are addressing in this paper.

Next, we investigate how different values of budget limit impact the number of events eventually saved to the cloud. We present the number of events saved at the cloud each day from May 1st, 2012 till July 31st, 2012. with different values of budget limit in Fig. 6. From the description of the sampling protocol, we can infer that the final number of saved events is not necessarily equal to the budget limit. Fig. 6 shows that this sampling protocol utilizes approximately 75% of the budget on average for different budget values. In Fig. 6, we also demonstrate that this sampling protocol works correctly in the presence of drastic changes, as the number of events monitored soars on the 40th day. In this case, the sampling protocol does not violate the budget constraints.

The underlying foundation of our sampling protocol is that uniformly sampling is ensured. We will see the importance of uniformity in one real application. The temperature sensor periodically measures the environmental temperature and sends the sensing data to the cloud for archiving purpose.

10 RELATED WORK

Wangetal. Rely on erasure correcting code to ensure the security of cloud data storage. They compute homomorphism tokens for data blocks dispersed across distributed servers and use a challenge-response protocol between users and the cloud to verify cloud data correctness. However, their scheme is not practical for signing sensing data samples. The set of data blocks that can be verified is predicated before data distribution while data applications determine which part of data to use in real time. The other reason is that the verification process can only be conducted by data owners instead of people who use the data. Their follow up work enables a third-party auditor to be a delegate for users to conduct cloud data verification. Their solution bases on Merkle Hash Tree. Our proposed signature scheme, GSC, achieves higher throughput with less overhead. GSC can be applied to a public audit to periodically check integration and authentication of cloud data storage.

11 DISCUSSION AND FUTURE WORK

Different sensing devices may send generated data to the coordinator at vastly different speed. The video surveillance system [17] continuously generates tons of data whereas human- motion detector [38] sends much less data occasionally. In order to avoid starvation of devices, the sampling protocol discussed in this paper can be easily generalized to allow weighted items. How to automatically set weights for different devices with little human intervention would be our future work.

Our sampling and signature scheme can be also applied to other areas besides IoT data storage. It is increasing important to monitor networks at different geographic locations in a scalable way [19]. Our sampling and signature scheme provides the opportunity to relieve the burden of the network, where the local collector acts as the coordinator and periodically transmits the sampled packets to the global traffic analytic. Since the sampled packets may be transmitted over the Internet [19], DTC is no longer applicable in this scenario where network switches send data at 10/40 gbps. The memory of switches cannot sustain to store such tremendous amount of internal nodes of authentication tree in DTC.

In this paper, we do not discuss the network issues associated with the sampling protocol, in which it is assumed that the communication between the coordinator and the device is instantaneous. Even though the network latency does not impact the

eventual correctness of the sampling protocol, a lot of network bandwidth is wasted owe to the transmission of events that should be discarded at devices locally. In our future work, we plan to design a queuing principle which prioritizes the coordinating messages to favor the devices sending more events thus to reduce the network bandwidth waste.

Fog computing is a new paradigm where small-scale cloud data centers are deployed at ISP network edge. Because their proximity to end-users, VMs of fog computing can be leveraged as the coordinator described in the sampling protocol. In this way, the end users are not bothered to update their access point to support sampling report.

12 CONCLUSION

We summarize the new challenges of the IoT data communication framework with authenticity and integrity and argue that existing solutions cannot be easily adopted. We design a system aimed to address these challenges. This system is able to uniformly sample data from sensing devices and then securely store the data in the cloud while respecting resource budget constraint. The sub-systems in our paper symbiotically operate together and this system is efficient in terms of space and time, as is validated by extensive simulation and prototype emulation experiments.

13 ACKNOWLEDGMENTS

The authors are supported by University of California Santa Cruz startup funding and National Science Foundation Grant CNS-1701681. The authors also thank anonymous IoT DI reviews for their constructive comments and suggestions.

14 REFERENCES

[1] 1992. The MD5 Message-Digest Algorithm. https://tools.ietf.org/html/rfc1321. (1992).
[2] 2011. mHealth: New horizons for health through mobile technologies. http://www.who.int/goe/publications/goe mhealth web. pdf. (2011).
[3] 2013. http://traces.cs.umass.edu/index.php/Smart/Smart. (2013).
[4] 2013. DSA. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS. 186-4.pdf. (2013).
[5] 2013. The Seagate 600 & 600 Pro SSD Review. http://www.anandtech.com/show/6935/seagate-600-ssd-review/5. (2013).
[6] 2014. Sampling for Big Data. www.kdd.org/kdd2014/tutorials / t10 part1.pptx. (2014).
[7] 2014. Samsung SSD 850 EVO 120GB, 250GB, 500G-B & 1TB Review. http://www.anandtech.com/show/8747/ samsung-ssd-850-evo-review/8. (2014).
[8] 2015. http://www.crn.com/slide-shows/cloud/300077635/ the-10-biggest-cloud-outages-of-2015-so-far.htm/pgno/0/2. (2015).
[9] 2015. HVAC Monitoring, Energy Monitoring and Control. http: //goo.gl/Ybq8Ai. (2015).
[10] 2015. SHA-1. http://csrc.nist.gov/publications/ftps/ftps180-4/ ftps-180-4.pdf. (2015).
[11]n.d.. https://www.opensensors.io/. (n.d.).
[12] n.d.. Amazon S3 Pricing. https://aws.amazon.com/s3/pricing/. (n.d.).
[13]n.d.. eHealth. http://www.who.int/topics/ehealth/en/. (n.d.). [14]n.d.. Nest. https://goo.gl/7uMdA1. (n.d.).
[15]n.d.. OpenSSL. https://www.openssl.org/. (n.d.).
[16]n.d.. xxHash. http://www.xxhash.com/. (n.d.).
[17] A. J. Brush, J. Jung, R. Mahajan, and F. Martinez. 2013. Digital neighborhood watch: Investigating the sharing of camera data amongst neighbors. In *Proc. of ACM CSCW*.
[18] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. 2012. Continuous sampling from distributed streams. *JACM* 59, 2 (2012).
[19] L. Elsen, F. Kohn, C. Decker, and R. Wattenhofer. 2015. goProbe: a scalable distributed network monitoring solution. In *Proc. of IEEE P2P*.
[20] D. Evan. 2011. The Internet of Things, Cisco White Pa- per. https://www.cisco.com/c/dam/en us/about/ac79/docs/ innov/IoT IBSG 0411FINAL.pdf. (2011).
[21] R. Gennaro and P. Rohatgi. 1997. How to sign digital streams. In *Crypto*.
[22] M. Gerla, E. Lee, G. Pau, and U. Lee. 2014. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *Proc. of IEEE WF-IoT*.
[23] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future tteneration Computer Systems* 29, 7 (2013).
[24] T. Gupta, R. P. Singh, A. Phanishayee, J. Jung, and R. Mahajan. 2014. Bolt: Data management for connected homes. In *Proc. of USEIX NSDI*.
[25] Y. Kim, J. Kang, D. Kim, E. Kim, P. K. Chong, and S. Seo. 2008. Design of a fence surveillance system based on wireless sensor networks. In *Proc. of the Autonomics*.
[26] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong. 2016. Privacy- Preserving Public Auditing Protocol for Low-Performance End Devices in Cloud. *IEEE Transactions on Information Forensics and Security* 11, 11 (2016).
Prince Mahajan, Srinath Setty, Sangmin Lee, Allen Clement, Lorenzo Alvisi, Mike Dahlin, and Michael Walftsh. 2011. Depot: Cloud Storage with Minimal Trust. *ACM Trans. Comput. Syst.* 29, 4, Article 12 (Dec. 2011), 38 pages. DOI:http://dx.doi.org/10.1145/2063509.2063512
[27] R. C. Merkle. 1987. A digital signature based on a conventional encryption function. In *Proc. of CRYPTO*.
[29]S. Monterde. 2014. Cisco Technology Radar. (2014).
[30] K. Piotrowski, P. Langendoerfer, and S. Peter. 2006. How public key cryptography influences wireless sensor node lifetime. In *Proc of ACM SASN*.
[31] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for

obtaining digital signatures and public-key cryptosystems. *CACM* 21, 2 (1978).

[32] J. Scott, Bernheim B., J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar. 2011. PreHeat: controlling home heating using occupancy prediction. In *Proc. of ACM Ubicomp*.

[33] J. S. Vitter. 1985. Random sampling with a reservoir. *ACM Trans. Math. Software* 11, 1 (1985).

[34] C. Wang, Q. Wang, K. Ren, and W. Lou. 2009. Ensuring Data Storage Security in Cloud Computing. In *Proc. of IEEE IWQoS*.

[35] C. Wang, Q. Wang, K. Ren, and W. Lou. 2010. Privacy-preserving public auditing for data storage security in cloud computing. In *Proc. of IEEE INFOCOM*.

[36] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. 2011. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE transactions on parallel and distributed systems* 22, 5 (2011), 847–859.

[37] C. K. Wong and S. S. Lam. 1998. Digital signatures for flows and multicasts. In *Proc. of IEEE ICNP*.

[38] T. Yamada, Y. Hayamizu, Y. Yamamoto, Y. Yomogida, A. Izadi- Najafabadi, D. N. Futaba, and K. Hata. 2011. A stretchable carbon nanotube strain sensor for human-motion detection. *Na- ture nanotechnology* 6, 5 (2011).

[39] Y. Zhang, L. Duan, and J. L. Chen. 2014. Event-driven soa for iot services. In *Proc. of IEEE SCC*.