Clouds of Things need Information Flow Control with Hardware Roots of Trust

Thomas F. J.-M. Pasquier, Jatinder Singh, Jean Bacon
Computer Laboratory, University of Cambridge
Email: firstname.lastname@cl.cam.ac.uk

Abstract—**There is a clear, outstanding need for new security mechanisms that allow data to be managed and controlled within the cloud-enabled Internet of Things. Towards this, we propose an approach based on *Information Flow Control* (IFC) that allows: (1) the continuous, end-to-end enforcement of data flow policy, and (2) the generation of provenance-like audit logs to demonstrate policy adherence and contractual/regulatory compliance. Further, we discuss the role of Trusted Platform Modules (TPMs) in supporting such a system, by providing *hardware roots of trust*. TPMs can be leveraged to validate software configurations, including the IFC enforcement mechanism, both in the cloud and externally via remote attestation.**

Keywords—Internet of Things, Information Flow Control, Provenance, Hardware Roots of Trust, Remote Attestation

I. Introduction

Internet of Things (IoT) solutions tend to rely heavily on cloud computing. In a recent survey [1], 33 out of 38 of the IoT infrastructures surveyed involved the use of cloud services. Traditionally, IoT applications were built in silos where sensors and actuators, cloud services and end-user applications were tightly coupled and inseparable. Current research and standardisation efforts aim towards breaking those silos, by allowing fully customisable service chains [2].

However, the challenges introduced by connected devices, potentially monitoring every aspect of the environment and daily life, require security beyond the current standard authentication, access control and secure channels [3]. There is a clear need for an end-to-end security mechanism that underlies every data exchange, and ensures that data is used for the purposes specified by those who own and/or are responsible for it [4].

We propose using *Decentralised Information Flow Control* (DIFC) [5], that extends traditional IFC [6], to address the security and privacy concerns inherent in IoT. This is by allowing the continuous management of data as it flows throughout IoT systems, including within and between cloud services. IFC technology has has been used for embedded software in BMW cars [7], has been proposed to control data usage of third party applications in a social network environment [8], and has been demonstrated to be applicable to more general use cases [9]. Our own implementation, CamFlow [10], extends IFC with provenance-like features [11], by providing a directed graph audit mechanism, to allow demonstration of compliance with policy (regulations and contracts).

In this paper we propose hardware-backed DIFC to complement the more established security mechanisms. In short, DIFC makes it possible:

- to tightly bind the data's purpose and other properties to the data itself;
- to specify and enforce policy throughout the data's lifetime, be it the original data or derived data obtained through its processing, even in distributed systems;
- to enable auditing through standard graph analysis techniques;
- to provide extremely simple yet powerful lightweight policy that can be placed low in the software stack (in our approach [10] at the OS kernel level), reducing the amount of software stack that has to be trusted.

§III presents our assumptions on the IoT operating environment. In §II, we discuss how existing security technologies can be leveraged for IoT, and highlight the remaining problems. §IV outlines DIFC dataflow constraints and how they provide simple security primitives. §V describes our implementation in a cloud context. In §VI, we discuss how trusted hardware can be leveraged to extend IFC for IoT. Finally, in §VII, we summarise our progress to date and present major research directions that remain.

II. Background

We now briefly consider the extent to which the established security mechanisms, assumed to be available for deployment when and where appropriate, can be used to meet the challenges brought by the emerging IoT. We then highlight major remaining issues.

PKI: *Public Key Infrastructure* (PKI) is important in enabling a wide-scale security regime. PKI allows 'things' to have private keys and public key certificates, perhaps signed by a *certificate authority* linking them to their owners, who are also associated with certificates. X.509 certificates [12] can be used for *authentication/identification* and *authorisation* [13], are widely available and well-understood. PKI technology is seen as important in enabling a wide-scale IoT infrastructure [14] and lightweight PKI schemes have been proposed (e.g. [15]) to support low-powered IoT devices.
Access control (AC): comprises authentication (you are who you claim) and authorisation (regulating whether an authenticated party may perform an action, e.g. reading data). Authentication builds on system-wide identification, where a certificate-based (PKI) model can be used. Authorisation policy is often role-based or embodied in access control lists.
Encryption: We assume the possibility for encrypted communication channels e.g. via TLS, or more lightweight versions for resource-challenged 'things'. TLS relies on PKI and certificate authorities. Application-level encryption can secure data beyond the application's scope, but this raises issues relating

to key management and distribution [3]. At present, processing generally cannot be carried out on encrypted data, which can, for instance, preclude some cloud-based services [4].

These security mechanisms can be used, or adapted to support IoT [3]. However, they tend to operate at specific policy enforcement points, concerning a particular principal taking a particular *action*. While this protects the specific interaction, it does not provide continuous control over the data. The approaches do not allow a data producer to control, or trace, their data after it has been transferred to a third-party.

This results in a great deal of trust being placed throughout the entire IoT supply chain—which includes cloud service providers and other collaborative entities—that data will (only) be used and transferred in accordance with the purposes for which it was originally produced and/or shared [4].

For example, 'sticky' policy approaches such as in EnCoRe [16] propose enforcement of high-level policy in the cloud, and their approach could be adopted for the IoT. However, they aim at providing expressiveness at a higher level of abstraction and are implemented at that level. Furthermore, 'sticky' policy approaches generally assume the collaboration of the applications or that no communication channel exists outside of the enforcement points. This results in (i) an unnecessarily large trusted computing base (TCB); (ii) policy complex to enforce and potentially impractical on low-end devices and (iii) making it difficult to establish trust (applications need to be trusted to some extent).

We argue that there is a clear need to complement the existing security approaches with mechanisms that operate beyond the point-based interaction, applying continuously throughout the IoT infrastructure. Simple low-level, data-centric security primitives must be provided to ensure the enforcement of the data owner's requirements regarding how and where its data is used. Furthermore, this security mechanism should be agnostic to and separate from applications.

Such policies must account for *secrecy* concerns, such as privacy and confidentiality aspects, as well as *integrity* concerns, including data quality and authority. The policy must be consistently enforced, throughout the IoT supply chain, backed with strong evidence of policy adherence.

III. Trust Assumptions

Our work considers end-to-end DIFC enforcement and audit throughout the IoT service chain. Several assumptions underpin our work, which represent general operational concerns:

Trusted Platform Module (TPM): We assume that TPM [17] or vTPM [18] technology are available across the cloud components and their integrity can be remotely monitored [19]. Such technology is also becoming available on commodity hardware and mobile devices [20].

Hardware Integrity: We assume that the cloud provider has taken sufficient technical and non-technical measures to ensure that the hardware has not been tampered with (e.g. see CISCO vs NSA). The hardware integrity of edge devices (e.g. mobile phones, home automation devices etc.) is harder to guarantee. Protecting the hardware is likely to fall under the responsibility of the device/data owner. This may present challenges in situations where these are different parties.

Low-level software stack: We assume that the integrity of the low-level software stack is recorded and monitored. The low-level software stack here includes: BIOS, boot loader code and configuration, Options ROM, host platform configuration, etc. We assume that this integrity measurement is kept safe and cannot be tampered with. Tamper-proof hardware storage may help with keeping integrity measurements safe.

Cryptographic Security: We assume cryptographic functions to be secure and data exchange across machines to be encrypted. We assume that message integrity on exchanges between machines can be verified. Furthermore, data may be encrypted on disc to provide a further guarantee.

Physical Security: We assume that best practice is in place to restrict physical access to the hardware managed by cloud providers, or third parties managing the underlying infrastructure on their behalf. Physical access to edge devices (e.g. things, mobile devices, etc.) is harder to monitor and protect.

IV. DIFC as simple security primitives

DIFC considers two types of security guarantees. *Secrecy* policy broadly covers the usage/confidentiality of the data, restricting its propagation to components trusted to handle such types of data. For example, medical data may be restricted to approved devices and services, or data may be restricted to a certain geographical area [21]. *Integrity* broadly covers the assessment of some properties (e.g. quality, state, origin) associated with the data. For example, a certain accuracy may be required or a storage service may refuse the responsibility of handling confidential data 'in-clear' and may first require its encryption [4].

In DIFC, *entities*—including processes and data-holding objects e.g. files, key-value store entries, messages—are associated with *secrecy* (S) and *integrity* (I) labels, encapsulating the security policy. A *label* is a set of a tags, each *tag* representing a particular security concern. The current state of these labels is the entity's *security context*.

A flow of information $A \rightarrow B$ is safe if and only if:

$$A \rightarrow B, \text{ iff } \{S(A) \subseteq S(B) \wedge I(B) \subseteq I(A)\}$$

This rule is enforced *continuously, against every flow in the system*, unlike traditional access control where there is no subsequent control of disclosure after a principal has been authorised to access data.

For example, a patient Alice may be discharged from hospital to home monitoring. Home monitoring data tagged as medical in its secrecy label S can only flow to a processing entity that contains a secrecy tag medical in its S. This ensures that data produced by Alice's home monitoring equipment is only processed by a dedicated cloud application. Further, selected medical data may be released for research. Such data should be secrecy tagged with the research purpose, e.g. diabetes-research, to ensure proper data usage. Further, entities running in the research environment may want to specify an integrity tag consent-to-research as their requirement for receiving and taking responsibility for the data [4].

The flow constraints have implications for the end-to-end properties of data throughout a chai of services. Secrecy and integrity tags are not only associated with some original data,

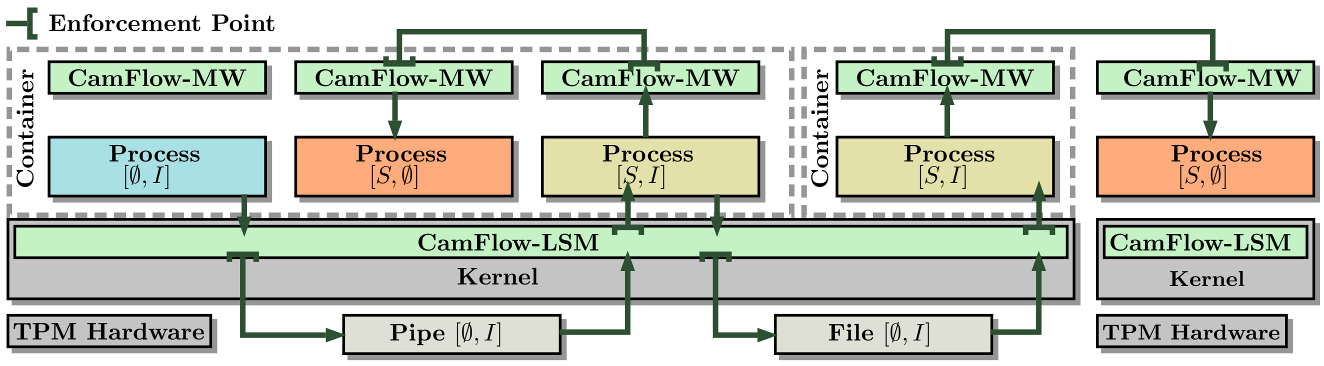

Fig. 1. CamFlow Information Flow Control enforcement within and between containers and machines.

but with all data derived from it whether by computation or combination. In an IoT scenario, not only is the integrity of an actuation command ensured within the last hop to the device, but the flow rule for integrity ensures that any prior components in a chain leading to this command have at least the same integrity tags.

DIFC policies are *simple to express and enforce.* This is particularly important in an IoT context as low powered edge devices need to be able to enforce the policy. Further, trusted components are able to change the security context of entities; e.g. declassifying data after anonymisation or endorsing data after sanitisation. We have demonstrated [4] that, through the simple flow constraints, and trusted components declassifying and endorsing data, it is possible to build complex policies.

V. Enforcing DIFC

In our implementation [10], we enforce DIFC *within* devices or cloud VM through use of a Linux Security Module framework [22]. Our module (CamFlow-LSM) can be deployed on Linux systems (desktop, cloud servers, Android devices or be embedded). We enforce DIFC *between* machines through a specific messaging middleware (CamFlow-MW[1]) operates on *nix based systems (iOS, Linux, Android etc.), and thus can operate to manage the interactions between ‘things’ and cloud services. Fig. 1 gives a general overview of our system architecture.

In this context an IFC-labelled active entity may be a cloud application instance, a mobile application, a process on a desktop OS etc. An entity’s local flow of data (through files, pipes, shared memory) is locally constrained. The LSM provides security hooks on interactions between kernel objects, which is where DIFC policies are enforced. External flows to and from components running under DIFC constraints are strictly restricted to the messaging MW which ensures security across machines: including secure channels and component authentication/authorisation. Software integrity verification is also possible through hardware features (see §VI).

Furthermore, *every data flow in the system is logged*. Metadata about the different interacting components and objects is logged and can be interpreted and analysed as a directed graph providing provenance-like features [11]. These audit logs allow compliance to be demonstrated, system behaviour to be understood and verification that policy is properly expressed and enforced. We give an example of compliance demonstration in a legal medical context in [4].

We assume that the rest of the kernel can be trusted and does not interfere with the enforcement mechanism. A further guarantee of this particular point could be leveraged from *hardware inverse sandbox* [23]. LSM system hooks have been statically and dynamically verified [24]–[26], and our implementation inherits from LSM the formal assurance of IFC’s correct placement on the path to any controlled kernel object. This is sufficient to guarantee that we control flow and record audit on any operation on a controlled kernel object.

Further, it has been demonstrated [27] that applications running on top of an IFC-enforcing OS need not be trusted. This indicates a security mechanism with an extremely thin TCB compared to those discussed in §II.

VI. Leveraging Trust from Hardware

DIFC protection is only guaranteed above the technical layer in which the DIFC mechanism operates. Safe exchange of data in a DIFC-context relies on trust placed in this mechanism. In a cloud context, the enforcement mechanism is provided and guaranteed by the cloud provider. If trust can be established with the cloud provider, no other party need to be trusted to guarantee secrecy and integrity of data [10]. This trust relationship can be established as the cloud provider is bound by contract, regulation and economic interest. This trust is demonstrated every day by companies adopting the cloud. However, when moving towards a potentially self-managed ‘things’ infrastructure, this trust relationship becomes more complex to establish. There is a need to demonstrate, reliably, that a particular machine has the appropriate untampered DIFC enforcement mechanisms in place.

One such approach is to leverage Trusted Platform Modules (TPM) [17], as used for remote attestation [28]. TPM is used to generate an unforgeable hash representing the state of the hardware and software of a given platform, that can be remotely verified. Therefore, a company could audit the implementation of a DIFC enforcement mechanism and ensure that the kernel security module, messaging middleware and the configuration they provide are indeed running on the platform. Any difference between the expected state of the software

[1]Our MW provides: message queues, service discovery, dynamic reconfiguration.

stack and the platform would be detected and might represent a breach of trust.

TPM, with remote attestation, is reaching maturity for cloud computing [18], with IBM rolling out an open source, scalable trusted platform based on virtual TPM [29]. Berger et al. [29] describe a mechanism allowing TPM and remote attestation to be provided for virtual machine and container-based solutions, covering the whole range of contemporary cloud offerings. Furthermore, the approach not only allows the state of the software stack to be verified at boot time, but also during execution, and can thus prevent run-time modification of the system configuration. Similar mechanisms exist for mobile phones [20] and embedded systems [30].

In addition to verifying a platform, the same techniques can also be used to verify the integrity of a remote system at run-time, to ensure that an entity has in place the appropriate IFC enforcement mechanisms before data is exchanged. This can be achieved through standard remote attestation techniques. Furthermore, integrity tags can be derived from a hardware-backed source, such as GPS-based location, using the mechanism described in [31].

VII. Conclusion

Data management remains a challenge for cloud computing, and these issues will be exacerbated by the emerging Internet of Things. In this paper, we presented our ongoing work on DIFC, as a mechanism for managing data within and throughout cloud-supported IoT infrastructure.

We described how our approach operates to complement existing security techniques, by enabling data management policy to be enforced continuously, system-wide. DIFC offers compelling advantages, but only if the DIFC implementation is itself trustworthy. Our key contribution here, is in considering the role of trusted hardware and remote attestation, which when combined with a DIFC implementation, can raise the levels of trust and assurance in the supporting infrastructure.

By using trusted hardware, remote attestation and a provenance-like audit log generated during DIFC enforcement, we are able to produce audit data that can be used to demonstrate compliance with regulatory or contractual obligations.

Acknowledgement

This work was supported by UK Engineering and Physical Sciences Research Council grant EP/K011510 CloudSafetyNet. We acknowledge the support of Microsoft through the Microsoft Cloud Computing Research Centre.

References

[1] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A Gap Analysis of Internet-of-Things Platforms," 2015, Arxiv, arXiv:1502.01181. [Online]. Available: http://arxiv.org/abs/1502.01181

[2] "Internet of Things (Preliminary Report 2014)," ISO/IEC JTC 1, Tech. Rep., 2015.

[3] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty security considerations for cloud-supported Internet of Things," *IEEE Internet of Things Journal*, 2015.

[4] J. Singh, J. Powles, T. Pasquier, and J. Bacon, "Data Flow Management and Compliance in Cloud Computing," *IEEE Cloud Computing Magazine, SI on Legal Clouds*, 2015.

[5] A. Myers and B. Liskov, "Protecting privacy using the decentralized label model," vol. 9, no. 4. ACM, 2000, pp. 410–442.

[6] D. E. Denning, "A lattice model of secure information flow," *Communication of the ACM*, vol. 19, no. 5, pp. 236–243, 1976.

[7] A. Bouard, B. Weyl, and C. Eckert, "Practical information-flow aware middleware for in-car communication," in *Workshop on Security, privacy & dependability for cyber vehicles*. ACM, 2013, pp. 3–8.

[8] K. Singh, S. Bhola, and W. Lee, "xBook: Redesigning Privacy Control in Social Networking Platforms," in *Security Symposium*. USENIX, 2009, pp. 249–266.

[9] N. Kumar and R. Shyamasundar, "Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels," in *Big Data and Cloud Computing (BdCloud'14)*. IEEE, 2014, pp. 753–760.

[10] T. Pasquier, J. Singh, D. Eyers, and J. Bacon, "CamFlow: Managed Data-Sharing for Cloud Services," *IEEE Transactions on Cloud Computing*, 2015.

[11] L. Carata, S. Akoush, N. Balakrishnan, T. Bytheway, R. Sohan, M. Selter, and A. Hopper, "A primer on provenance," *Communications of the ACM*, vol. 57, no. 5, pp. 52–60, 2014.

[12] S. Farrell and R. Housley, "An Internet Attribute Certificate Profile for Authorization," IETF, Tech. Rep., 2002.

[13] D. W. Chadwick, A. Otenko, and E. Ball, "Role-based Access Control with X. 509 Attribute Certificates," *Internet Computing, IEEE*, vol. 7, no. 2, pp. 62–69, 2003.

[14] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *Internet of Things Journal, IEEE*, vol. 1, no. 3, pp. 265–275, 2014.

[15] J. Granjal, E. Monteiro, and J. S. Silva, "End-to-end Transport-layer Security for Internet-integrated Sensing Applications with Mutual and Delegated ECC Public-Key Authentication," in *IFIP Networking Conference, 2013*. IEEE, 2013, pp. 1–9.

[16] S. Pearson and M. C. Mont, "Sticky Policies: An Approach for Managing Privacy across Multiple Parties," *Computer*, vol. 44, July 2011.

[17] T. Morris, "Trusted Platform Module," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 1332–1335.

[18] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the Trusted Platform Module," in *Security Symposium*. USENIX, 2006, pp. 305–320.

[19] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang, "Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence," in *Dependable Systems & Networks (DSN'09)*. IEEE, 2009, pp. 115–124.

[20] M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert, "Beyond Kernel-level Integrity Measurement: Enabling Remote Attestation for the Android Platform," in *Trust and Trustworthy Computing*. Springer, 2010, pp. 1–15.

[21] T. Pasquier and J. Powles, "Expressing and Enforcing Location Requirements in the Cloud using Information Flow Control," in *IC2E International Workshop on Legal and Technical Issues in Cloud Computing (Claw'15)*. IEEE, 2015.

[22] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman, "Linux Security Modules: General security support for the Linux kernel," in *Foundations of Intrusion Tolerant Systems*. IEEE, 2003, pp. 213–213.

[23] "Software Guard Extensions Programming Reference," Intel, Tech. Rep. 329298-001US, 2013, accessed: 29th September 2015. [Online]. Available: https://software.intel.com/sites/default/files/329298-001.pdf

[24] A. Edwards, T. Jaeger, and X. Zhang, "Runtime verification of authorization hook placement for the Linux Security Modules framework," in *Conference on Computer and Communications Security*. ACM, 2002, pp. 225–234.

[25] T. Jaeger, A. Edwards, and X. Zhang, "Consistency analysis of authorization hook placement in the Linux security modules framework," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 2, pp. 175–205, 2004.

[26] V. Ganapathy, T. Jaeger, and S. Jha, "Automatic placement of authorization hooks in the Linux security modules framework," in *Conference on Computer and Communications Security*. ACM, 2005, pp. 330–339.

[27] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris, "Information Flow Control for Standard OS Abstractions," in *Symposium on Operating Systems Principles*. ACM, 2007, pp. 321–334.

[28] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," in *Conference on Hot Topics in Cloud Computing*. USENIX, 2009, pp. 3–3.

[29] S. Berger, K. Goldman, D. Pendarakis, D. Safford, E. Valdez, and M. Zohar, "Scalable Attestation: A Step Toward Secure and Trusted Clouds," in *International Conference on Cloud Engineering (IC2E)*. IEEE, 2015.

[30] N. Aaraj, A. Raghunathan, and N. K. Jha, "Analysis and Design of a Hardware/Software Trusted Platform Module for Embedded Systems," *Transactions on Embedded Computing Systems (TECS)*, vol. 8, no. 1, p. 8, 2008.

[31] K. R. Jayaram, D. Safford, U. Sharma, V. Naik, D. Pendarakis, and S. Tao, "Trustworthy Geographically Fenced Hybrid Clouds," in *ACM/IFIP/USENIX Middleware*. ACM, 2014.