# Information Flow Control with Minimal Tag Disclosure

Hajoon Ko
University of Cambridge
first.last@cl.cam.ac.uk

Jatinder Singh
University of Cambridge
first.last@cl.cam.ac.uk

Thomas F. J.-M. Pasquier
University of Cambridge
first.last@cl.cam.ac.uk

Changyu Dong
University of Strathclyde
first.last@strath.ac.uk

David Eyers
University of Otago
dme@cs.otago.ac.nz

Jean Bacon
University of Cambridge
first.last@cl.cam.ac.uk

## ABSTRACT

Information Flow Control (IFC) extends conventional access control beyond application boundaries, and allows control of data flows after a point of authorised data disclosure. In a deployment of IFC within a cloud operating system (OS), the IFC implementation can be trusted by applications running over the same OS instance. In an IFC deployment within a widely distributed system, such as in the Internet of Things, the potential for trustworthy enforcement of IFC must be ascertained during connection establishment. IFC is based on tagging data in line with data management requirements. When audit is included as part of IFC, it can be shown that a system complies with these requirements.

In this paper, we consider the possibility that some tags may be sensitive and discuss the use of *Private Set Intersection (PSI)* to prevent unnecessary disclosure of IFC tags during the establishment of communication channels. The proposed approach guarantees that on authorised flows, only the tags necessary for that interaction are disclosed and that no tags are disclosed for prevented flows. This functionality is particularly important in contexts such as healthcare, where privacy and confidentiality are paramount.

## CCS Concepts

•**Security and privacy → Information flow control;** *Distributed systems security;* Public key (asymmetric) techniques;

## Keywords

Information Flow Control, Private Set Intersection

## 1. INTRODUCTION

Our previous work [12] explores Information Flow Control (IFC) as an information-centric security mechanism to ensure safe sharing of data across application boundaries in a cloud computing context. This is unlike traditional access control mechanisms, where the data owner often loses control over their data as it moves beyond the boundaries of the applications. Indeed, IFC allows the sharing of data between mutually distrusting parties [6], as long as the underlying enforcement mechanism is mutually trusted. IFC can guarantee that shared data, or data derived from shared data, cannot be used outside of a specified purpose [7]. This is achieved by attaching labels (each comprising a set of tags) to all entities within the IFC-protected system. Flows are prevented unless the labels accord, as described in §2.

Our earlier work relies on the assumption that there is a single IFC-enforcing party, the PaaS cloud provider. This cloud provider is incentivised to guarantee the proper enforcement of the security mechanism as part of their service offering. When IFC is extended to a wider distributed environment, such as for the Internet of Things (IoT), challenges arise given the multitude of players involved.

In our current implementation, a communication channel between two applications is established after mutual authorisation and verification of their IFC labels. Initial thoughts on how to ensure a remote party, in an IoT context, is subject to a trustworthy enforcement regime is discussed in [13].

In this paper we focus on managing the fact that IFC tags may be sensitive.

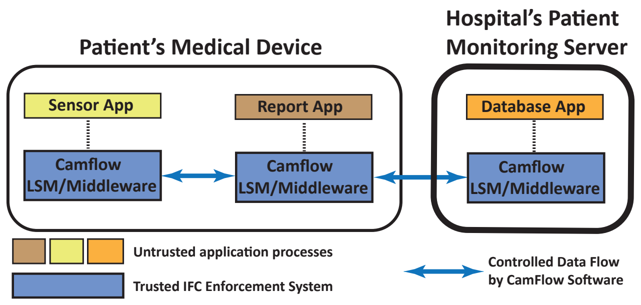
Consider a self-monitoring patient scenario where both a hospital-issued medical device and a patient's own fitness/wellbeing "app" feed data into the patient's medical record. The patient's medical record software may have sensitive tags that should not be disclosed to the app on the patient's mobile phone. Our current implementation was developed for a trusted environment, where IFC is transparent to application instances and, at present, all tags are disclosed as part of establishing a communication channel. However, it is generally preferable that tags are disclosed on a *"need-to-know"* basis.

We propose to leverage well known techniques to extend IFC to a non-uniformly trusted, distributed world (IoT). We use X.509 certificates for authorisation and tag verification, as described in [18], using attribute certificates to represent tags [1, 3]. *Private Set Intersection* (PSI) [5] is a cryptographic mechanism that allows two parties to compute the intersection of two sets without revealing the elements outside of the intersection. In this paper, we present a protocol that uses PSI to prevent the unnecessary disclosure of tags between communicating parties, as IFC is enforced on connection establishment.

*International Conference on Internet of Things and Cloud Computing '16 Cambridge, United Kingdom*

© 2016 ACM. ISBN 978-1-4503-4063-2…$15.00

DOI: 10.475/123_4


Hospital's Patient Monitoring Server
Patient's Medical Device
Sensor App
Report App
Database App
Camflow LSM/Middleware
Camflow LSM/Middleware
Camflow LSM/Middleware
Untrusted application processes
Trusted IFC Enforcement System
Controlled Data Flow by CamFlow Software
Figure 1: CamFlow Architecture.

## 2. CAMFLOW IFC IMPLEMENTATION

IFC is not a replacement for existing security mechanisms such as traditional, point-based access controls or the encryption of sensitive data, but rather is complementary, providing guarantees when sharing sensitive data between parties or services. IFC specifies the trust assumptions that need to be made. Instead of a need to trust all parties involved, only trust in the IFC enforcement mechanism is required.

Our model (a full description is available in [12]) derives from Myers' model [8] where data flow policy is encapsulated using secrecy and integrity *labels* associated with every entity of the considered system. These labels are each a set of *tags* with each tag representing a particular security concern. Security tags might include e.g. medical_data, alice_private, encrypted; integrity tags might include e.g. validated_input, standard_format, some_trusted_datasource.

$S(A)$ represents the set of secrecy tags and $I(A)$ the integrity tags held by an entity $A$. Data flow between an entity $A$ and an entity $B$ is allowed if and only if:

$$S(A) \subseteq S(B) \ \wedge \ I(B) \subseteq I(A)$$

In other words, data flow is allowed if and only if $A$'s secrecy label is a subset of $B$'s and $B$'s integrity label is a subset of $A$'s. The *security context* of an entity is defined as the state of its $S$ and $I$ labels.

IFC is a data-centric mandatory access control mechanism that guarantees non-interference across security contexts. The security properties of interacting entities are checked continuously on every data exchange.

CamFlow [12] is a cloud-targeted implementation of IFC. A Linux Security Module (LSM) enforces IFC constraints between kernel objects (processes, files, pipes, etc.) on the local machine, while a communication middleware is used for IFC-constrained, inter-machine communication [16]. We illustrate our architecture in Fig. 1.

To move towards a distributed environment such as IoT, we described in [18] how X.509 standard mechanisms can be used to represent IFC concepts. Public Key Certificates are used to uniquely identify entities in the system; they are referred to as Identity Certificates (IDC). Attribute Certificates (AC) [3] are used to represent IFC tags, and are attributes linked to the identified entity. In a CamFlow context, IDCs are used to authenticate applications (e.g. Bob's medical smartphone app). After authentication, ACs are exchanged between parties (e.g. Bob's app and a web application instance on the hospital server) to determine, based on IFC constraints, whether the communication should be authorised.

Our current IFC implementation, which is developed for PaaS clouds, assumes trustworthy IFC enforcement. Enforcement is transparent to application instances; application managers allocate tags to application instances, without the application's direct involvement. Establishing a connection between applications occurs though a middleware. This process involves mutual endpoint authorisation, after which IFC enforcement takes place [12]. Currently, the communicating entities' IFC components reveal their entire sets of tags to each other to check the IFC subset relationship, after which the flow is accepted or rejected.

As motivated in §1, some tags may be sensitive and a means of enforcing IFC without disclosing all the parties' tags is desirable. As a general principle, tags should only be disclosed on a "need-to-know" basis, and especially for the potentially many parties dynamically involved in IoT chains. The next section describes a means for establishing common subsets of tags between entities: private set intersection (PSI). Incorporating PSI into IFC enforcement means that if no flow is allowed, neither party knows any tags of the other party. If communication is allowed, the sending party knows only that its secrecy tags are a subset of the receiver's and the receiving party that its integrity tags are a subset of the sender's.

## 3. IFC WITH PSI FOR IoT

PSI enables two parties, $A$ and $B$, holding a set of elements $S_A$ and $S_B$ respectively, to compute the intersection $S_A \cap S_B$ without revealing to each other any other non-intersecting elements between their sets. PSI is an active research area and has been applied to many privacy-sensitive applications. For example, a government agency and an airline could perform PSI to identify terrorist suspects on an airline's passenger list, without directly revealing the full list of suspects or passengers to each other [2].

To explain the application of PSI in an IFC context, we consider a scenario where a fitness/wellbeing device (e.g. fitbit) – hereafter $A$ – is used to transmit patient activity data to a hospital service – hereafter $B$.

To establish a one-way communication channel from $A$ to $B$ ($A \rightarrow B$) to satisfy the IFC constraints introduced in §2, $A$ must verify that $S(A) \subseteq S(B)$, that is $S(A) = S(A) \cap S(B)$. $B$ must verify $I(B) \subseteq I(A)$, that is $I(B) = I(A) \cap I(B)$.[1]

Fig. 2 shows how a generic PSI scheme can be leveraged to enforce IFC without disclosing unnecessary tags to either party. In our example, that means Alice's device $A$ such that $S(A) = \{alice\}$ can only verify if Alice's hospital service instance $B$ runs with the label $S(B) = \{alice\}$, without $A$ learning about any other of $B$'s potential tags such as *medical* or *AIDS*.

For evaluation of IFC with PSI, we selected Oblivious Pseudo-Random Functions (OPRF) [2]. The basic idea is to hash the tags before transmitting them for comparison. Since simple hashing of well-known tag names is subject to guessing (e.g. dictionary attacks), public-key cryptography is used to prevent this (see [2] for details).

[1]To establish two-way communication both parties must independently verify $S(A) = S(B)$ and $I(A) = I(B)$, which can be efficiently achieved through a private equality test as for the example described in [4].

– A's secrecy tags : $\mathcal{S_A} = \{sa_1, sa_2, ...sa_{|\mathcal{S_A}|}\}$
– A's integrity tags : $\mathcal{I_A} = \{ia_1, ia_2, ...ia_{|\mathcal{I_A}|}\}$
– B's secrecy tags : $\mathcal{S_B} = \{sb_1, sb_2, ...sb_{|\mathcal{S_B}|}\}$
– B's secret input : $\mathcal{I_B} = \{ib_1, ib_2, ...ib_{|\mathcal{I_B}|}\}$

1) A $\xrightarrow{\mathcal{S_A}} \boxed{\text{PSI Protocol}} \xleftarrow{\mathcal{S_B}}$ B
   A $\xleftarrow{\mathcal{S_C}} \boxed{\text{PSI Protocol}} \xrightarrow{\mathcal{S_C}}$ B (where $\mathcal{S_C} = \mathcal{S_A} \cap \mathcal{S_B}$)

   A $\xrightarrow{\mathcal{I_A}} \boxed{\text{PSI Protocol}} \xleftarrow{\mathcal{I_B}}$ B
   A $\xleftarrow{\mathcal{I_C}} \boxed{\text{PSI Protocol}} \xrightarrow{\mathcal{I_C}}$ B (where $\mathcal{I_C} = \mathcal{I_A} \cap \mathcal{I_B}$)

2) Data flow A to B
   A : If $\mathcal{S_C} \neq \mathcal{S_A}$, disallow outgoing data.
   B : If $\mathcal{I_C} \neq \mathcal{I_B}$, disallow incoming data.

3) Data flow B to A
   B : If $\mathcal{S_C} \neq \mathcal{S_B}$, disallow outgoing data.
   A : If $\mathcal{I_C} \neq \mathcal{I_A}$, disallow incoming data.

Figure 2: PSI applied to IFC policy enforcement.

**Trust and Threat Model**

It is beyond the scope of this paper to consider attacks to PSI. Moreover, there are a number of outstanding issues in IoT that require attention [9,17].

Our concern in this paper is to reveal tags on a "need-to-know" basis between verified IFC implementations as general good practice. Our adversarial model is an "honest-but-curious" entity. A health professional's device labelled with $S(A) = \{medical\}$ can review and analyse general medical data $S(D) = \{medical\}$, or receive medical data from a remote process $S(P) = \{medical\}$. If $A$ attempts to read data labelled $S(D') = \{medical, Alice, AIDS\}$, IFC will prevent this flow, but $A$'s IFC enforcement component learns all the data's tags in the process. We are aiming to prevent such a potential leakage.

If IFC is to be used in an IoT context, the IFC enforcement mechanism must be trusted. TPM [10] is a physically tamper-resistant security chip that verifies the software stack on a given hardware platform at boot time [14] and monitoring is continued during execution. We assume the integrity of a remote IoT device's CamFlow software and its underlying platform can be verified by TPM technology [13].

**CamFlow IFC-PSI connection establishment**

The IFC-PSI protocol is described in Fig. 3. Communication between processes $A$ and $B$, whose communications are controlled by their IFC enforcement systems $CF_A$ and $CF_B$. The IFC implementation ensures that all external communication is via these components. Each application process has its own identity certificate ($IDC$) and one or more attribute certificates ($AC$s) specifying its assigned tags. When application processes communicate remotely, they invoke their local IFC middleware, $CF_A$ and $CF_B$ respectively.

In the setup stage (step 0), $CF_A$ and $CF_B$ load and verify the identity certificate ($IDC$) and tag certificates ($AC$s) of their application processes $A$ and $B$. In step 1, $A$ requests $CF_A$ for remote data transmission to $B$ whose external data flow is under the control of $CF_B$. In step 2, $CF_A$ and $CF_B$ authenticate each other based on Public Key Infrastructure (PKI), using their $IDC$s. In step 3, $CF_A$ and $CF_B$ run the PSI protocol by using the tags of their application processes as PSI inputs. In steps 4 and 5, $CF_A$ and $CF_B$ exchange $AC$s for their common secrecy and integrity tags to be verified by each other. In step 6, the connection between $A$ and $B$ is established and data can be transmitted.

**Definitions**

| | |
|---|---|
| - $\mathrm{CF_A}$, $\mathrm{CF_B}$ | : CamFlow IFC middleware enforcement process invoked by $A$, $B$ |
| - $IDC_A, IDC_B$ | : identity certificate issued to $A$, $B$ |
| - $A, B$ | : application processes whose data flow is controlled by $CF_A$, $CF_B$ |
| - $AC_{(A,t_i)}, AC_{(B,j)}$ | : $A, B$'s attribute certificate for tag $t_i$, $t_j$ |

**IFC-PSI Protocol**

| | | |
|---|---|---|
| 0) | $CF_A$ | : verify $A$'s $IDC$ and $AC$s prepare $\mathcal{S_A}$ and $\mathcal{I_A}$ |
| | $CF_B$ | : verify $B$'s $IDC$ and $AC$s prepare $\mathcal{S_B}$ and $\mathcal{I_B}$ |
| 1) | $A \to CF_A$ | : request a remote data transmission to $B$ |
| 2) | $CF_A \leftrightarrow CF_B$ | : PKI-based mutual IDC verification |
| 3) | $CF_A \leftrightarrow CF_B$ | : execution of a PSI protocol |
| 4) | $CF_A \to CF_B$ | : $\{AC_{(A,t)} \mid t \in \mathcal{I_C}\}$ |
| | $CF_B$ | : verify $A$'s $AC$s and its integrity tags |
| 5) | $CF_B \to CF_A$ | : $\{AC_{(B,t)} \mid t \in \mathcal{S_C}\}$ |
| | $CF_A$ | : verify $B$'s $AC$s and its secrecy tags |
| 6) | $[A, CF_A] \to [CF_B, B]$ | : data flow from $A$ to $B$ with IFC enforced by $CF_A$ and $CF_B$ |

Figure 3: The IFC-PSI protocol.

## 4. EVALUATION

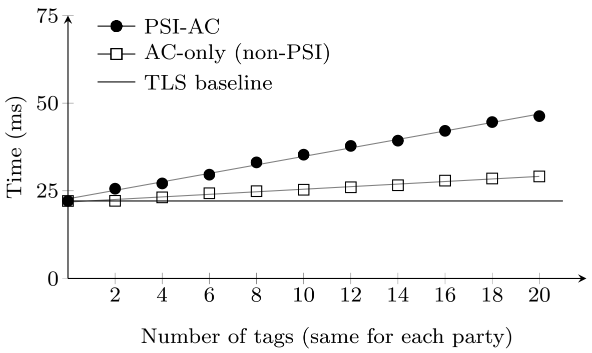


Figure 4: PSI overhead comparison.

To demonstrate the approach, we integrated our IFC-PSI protocol into SBUS [15] (CamFlow's middleware). The protocol operates as part of establishing a communication channel. For our implementation, we leveraged the OPRF-based PSI implementation by the SPROUT Laboratory at the University of California Irvine.[2] We also use the X.509 Attribute Certificate (AC) library, which is an OpenSSL wrapper developed by the Pervasive Computing group at the University Carlos III,[3] for general tag exchange and management.

[2]http://sprout.ics.uci.edu/
[3]http://www.it.uc3m.es/dds/index.html

| TLS | PSI | AC | Total |
|---|---|---|---|
| 22.15ms | 18.12ms | 6.03ms | 46.3ms |

Table 1: Delay breakdown for TLS, PSI and AC stages when each party holds 20 tags.

Our experiment involved two 1.6GHz Intel machines with 4GB RAM running Ubuntu 14.04 LTS, directly connected through a local 1000BASE-T Ethernet network. The results are presented in Fig. 4. In the graph, the horizontal line represents the (constant) overhead imposed by establishing a TLS connection. The AC-only scheme is where all tags are exchanged (and verified), while PSI-AC represents our 'minimal disclosure' approach. From the results, we see that PSI-AC imposes a steeper (linear) overhead than that of AC-only. This is because processing each PSI element requires non-negligible cryptographic computations.

Our experiment was to demonstrate the feasibility of a PSI-based tag exchange for IFC enforcement. Given the nature of the IFC model (§2), we see potential optimisations with respect to the subset and equality relationships between the tags held by each party. Further, in the IFC use cases we have considered to date, we have found that typically policy can be represented with only a few tags [19]. In Table 1 we present the overheads for a situation involving 20 tags, to indicate performance overheads in what is more of a worst-case scenario. Note also that PSI is an active area of research, and more efficient protocols may become available.

## 5. CONCLUSION

As the IoT evolves, we expect many end-user devices to connect with cloud-based services for data processing and archiving. Such services may handle sensitive data from many other sources and the IFC tags they hold for such purposes should not be freely revealed. In an IoT context, we must establish during connection set up that the IFC enforcement is trustworthy, i.e. is running hardware-verified IFC-enforcement software. Even so, it is good practice to reveal IFC tags only on a "need-to-know" basis so that sensitive tags are never disclosed beyond the domain in which they apply.

A full threat and risk model for IoT including cloud services is beyond the scope of this paper. IFC enables policy-specified control of where data flows, with audit of all allowed and rejected flows [11]. This creates the ability to demonstrate that services comply with their data management obligations.

In this paper we have argued that PSI could be a useful addition to IFC, so that only the tags that are essential to IFC-controlled communication are revealed to the parties concerned. We have shown that it is possible to incorporate a recent PSI protocol into our existing CamFlow (IFC) implementation. In future work, we see potential for optimisation and refinement.

## 6. ACKNOWLEDGEMENTS

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

## 7. REFERENCES

[1] D. W. Chadwick, A. Otenko, and E. Ball. Role-based Access Control with X. 509 Attribute Certificates. *Internet Computing, IEEE*, 7(2):62–69, 2003.

[2] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *14th International Conference on Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

[3] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. Technical report, IETF, 2002.

[4] M. Freedman, C. Hazay, K. Nissim, and B. Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, pages 1–41, 2014.

[5] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology-EUROCRYPT 2004*, pages 1–19. Springer, 2004.

[6] 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*, pages 321–334. ACM, 2007.

[7] N. Kumar and R. Shyamasundar. Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels. In *Big Data and Cloud Computing (BDCloud'14)*, pages 753–760. IEEE, 2014.

[8] A. C. Myers and B. Liskov. A Decentralized Model for Information Flow Control. In *Symposium on Operating Systems Principles (SOSP)*, pages 129–142. ACM, 1997.

[9] H. Ning, H. Liu, and L. Yang. Cyberentity Security in the Internet of Things. *Computer*, 46(4):46–53, April 2013.

[10] B. Parno. Bootstrapping Trust in a" Trusted" Platform. In *Conference on Hot Topics in Security (HotSec'08)*. USENIX, 2008.

[11] T. Pasquier, J. Singh, , J. Bacon, and D. Eyers. Information Flow Audit for PaaS clouds. In *International Conference on Cloud Engineering (IC2E)*. IEEE, 2016.

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

[13] T. F. J.-M. Pasquier, J. Singh, and J. Bacon. Clouds of Things need Information Flow Control with Hardware Roots of Trust. In *International Conference on Cloud Computing Technology and Science (CloudCom'15)*. IEEE, 2015.

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

[15] J. Singh, D. Eyers, and J. Bacon. Policy Enforcement within Emerging Distributed, Event-Based Systems. In *ACM Distributed Event-Based Systems (DEBS'14)*, pages 246–255, 2014.

[16] J. Singh, T. Pasquier, J. Bacon, and D. Eyers. Integrating Middleware and Information Flow Control. In *International Conference on Cloud Engineering (IC2E)*, pages 54–59. IEEE, 2015.

[17] 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.

[18] J. Singh, T. F. J.-M. Pasquier, and J. Bacon. Securing Tags to Control Information Flows within the Internet of Things. In *International Conference on Recent Advances in Internet of Things (RIoT'15)*. IEEE, 2015.

[19] 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.