PACCo: Privacy-friendly Access Control with Context

Andreas Put and Bart De Decker
KU Leuven, Dept. of Computer Science, iMinds-DistriNet, Celestijnenlaan 200A, 3001 Heverlee, Belgium

Keywords: Privacy, Context-aware Access Control, Policy Language.

Abstract: We propose a secure and privacy friendly way to strengthen authentication mechanisms of online services by taking context into account. The use of context, however, is often of a personal nature (e.g. location) and introduces privacy risks. Furthermore, some context sources can be spoofed, and hence, the level of trust of a verifier in a context source can vary.
In this paper, a policy language to express contextual constraints is proposed. In addition, a set of protocols to gather, verify and use contextual information in access control decisions is described. The system protects user privacy as service providers do not learn precise context information, and avoids linkabilities. Finally, we have implemented this system and our experimental evaluation shows that it is practical to use.

1 INTRODUCTION

During the last decade, the Internet landscape has changed from an information platform to a service platform. Almost all our daily needs can be satisfied by Internet services. Television, radio, shopping and socializing are just some examples where Internet services have become popular. Furthermore, smartphones have also become commonplace. These devices not only are more prevalent in today's society, they also are being used more and more to access online services. Now, a surge of other "smart"-devices can be observed. Consumer items like smart-watches, smart-cars or smart-lights are slowly gaining popularity. These items are often described as *Internet of Things* (IoT) appliances.

In order to make use of Internet services, users need to authenticate themselves to a service provider. Password-based authentication is the preferred way of authentication for these services. Not because of its security properties, but rather because of its superior usability properties compared to other authentication mechanisms (Riva et al., 2012). However, passwords are a weak authentication form as they can easily be guessed, shared or stolen (Adams and Sasse, 1999).

Context can be defined as "any information that can be used to characterize the situation of an entity. This entity is a person, place, or object considered relevant to the interaction between a user and an application, including the user and applications themselves" (Abowd et al., 1999). Contextual information can help to secure transactions without requiring user-interaction. For example, credit-card companies use the geolocation of a transaction to help detect fraud. Furthermore, smartphones and IoT-devices can capture a wide variety of contextual information. Some of this information, like location, proximity, or current activity can be used to assist in making access control decisions.

However, contextual information is often privacy intrusive. This problem is made even worse when contextual data from different transactions is compiled into a single profile. A study conducted by Groopman (Groopman, 2015) with more than 2000 participants clearly illustrates this problem. The study concludes that almost 80% of the participants are concerned about the information that service providers gather and analyze. Furthermore, using context to make authorization decisions introduces a security risk: context can sometimes be forged or spoofed.

In this paper, we propose PACCo, a system that focusses on the secure and privacy-friendly collection and usage of contextual information that can assist in making access control decisions. To do this, we introduce a new entity, the *Context Verifier* (CV), which will verify, certify, and, if possible, anonymize contextual information. Furthermore, our system is privacy-friendly, as multiple transactions with the CV cannot be linked, while the service provider learns the least amount of required context information. In addition, we define a policy language that focuses on expressing contextual requirements and also considers the security guarantees that different context sources can offer.

Put, A. and Decker, B.
PACCo: Privacy-friendly Access Control with Context.
DOI: 10.5220/0005969501590170
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT*, pages 159-170
ISBN: 978-989-758-196-0

This paper makes the following key contributions:

- We have combined existing privacy-enhancing technologies and cryptographic algorithms in PACCo's protocols, to allow a privacy-friendly and secure collection and usage of context. What kind of context is used to make access control decisions is not in this paper's scope.
- We propose a context-aware policy language.
- We have implemented our system and show our experimental results.

The paper is structured as follows: Section 2 presents related work and Section 3 illustrates motivational use cases, after which the preliminaries section follows. Next, we define PACCo's policy language. Section 6 shows an overview of PACCo, our threat model and the protocols, after which the security and privacy of PACCo are discussed. Finally, we show the test results of our prototype implementation.

2 RELATED WORK

XACML, eXtensible Access Control Markup Language version 3.0 was ratified by OASIS standard organization (Rissanen et al., 2013) in 2005. XACML defines a declarative access-control policy language defined in XML, and a processing model describing how to evaluate access requests according to the rules defined in policies. The XACML specification supports identity-based access control and incorporates some limited contextual information without any formal context-aware access control mode. Furthermore, geoXACML (Matheus and Herrmann, 2008) propose a set of extensions to introduce location constraints to the XACML standard.

A fair amount of work has been put into adding support for context into Role Based Access Control (RBAC) models. These models focus mostly on temporal and spacial context (Ray and Toahchoodee, 2007; Atluri and Chun, 2007), but also on support for more general context (Kulkarni and Tripathi, 2008; Bhatti et al., 2005). In (Hu and Weaver, 2004), a context-aware access control model for distributed health-care applications is proposed. Application designers determine which context types are to be used by analyzing the system's security requirements. Furthermore, (Ardagna et al., 2010) describes a context supporting policy model that focuses to better control "break the glass" attempts in health-care systems.

A general context-aware mandatory access control model is proposed in (Jafarian and Amini, 2015). This work is capable of dynamic adaptation of policies with context and it offers support for confidentiality and integrity requirements.

PACCo's goal is different from these systems, as PACCo also focuses on minimal information learning and secure collection and validation of the actual contextual information.

In Attribute-Based Access Control (ABAC) systems (Yuan and Tong, 2005; Vimercati et al., 2012; Jin et al., 2012), subjects are represented by a set of attributes, and access control decisions are made based the current value of these attributes. Context-aware access control is, in essence, similar to ABAC. However, here, the attributes of a subject are often dynamic in nature. Furthermore, contextual information can originate from a wide range of devices that are not necessarily under a verifier's control.

Context-aware authentication between a user and her device is another area that has been thoroughly explored (Riva et al., 2012; Hintze et al., 2015; Shebaro et al., 2015; Hayashi et al., 2013). However, privacy is of little concern to these systems as the context that is used does not leave the user's device. Furthermore, these papers do analyze how contextual information should be used (e.g. which context types to use or how to combine them to make a decision), whereas PACCo focusses on the secure and privacy-friendly collection and verification of context.

The CSAC system (Hulsebosch et al., 2005) does focus on offering context-aware online authentication with privacy. However, PACCo targets stricter privacy goals. In CSAC, users obtain location granting tickets from a context broker, which is in contact with different context providers. This ticket is presented to a service provider, which will use it to obtain the user's context at the broker. However, the context broker is able to learn everything about its users (their context, the services that they use, etc.), as it relays all the messages. PACCo avoids this by separating the interaction between the service provider and the context broker.

3 CONTEXT-AWARE AUTHENTICATION

We illustrate the benefits and problems caused by context-aware access control in two examples.

Conference Proceedings: The organizers of a scientific conference want to distribute the proceedings to the participants. To do so, the organizers make the proceedings available on an online file-server. The files are password-protected in order to allow only the

registered participants to download the proceedings. The organizers, however, believe that this is not secure enough, as the passwords can easily be shared.

Therefore, the organizers want to only allow access to participants who are present at the conference venue. In addition, access to the proceedings is allowed before the conference starts or closes if the participant is in the conference's city.

Contextual access control offers certainly new interesting possibilities. However, it also opens the door for privacy risks. The contextual information can be detailed or even unique if the location is determined by e.g. GPS. Furthermore, additional privacy concerns come into play when a third party takes care of the access control. This would require the participants to disclose their location data to a third party.

Consultant Access: In a company setting, an external consultant joins an in-house team on a project. Each Monday morning, the team holds a meeting to plan the following week. Furthermore, certain documents are accessible through an online portal, which is provided by a cloud service.

The consultant is able to access relevant documents for the meetings, but she is only able to do this during a meeting and if she is present in the meeting room. In addition, she is also granted access to those files, but only if she is in the vicinity of the project leader.

Security is essential in this scenario. Contextual information can be extremely useful for making access control decisions. However, it is important to note that relying on contextual information alone is often not sufficient. In addition, the information's integrity and authenticity should be verifiable and the freshness of the information plays an important role as well. Finally, the company can set up trusted devices that provide contextual information, but it might not want this information to leave the company. Therefore, it is important that the verification of raw context (e.g. by the company), and making the actual access control decision (e.g. by the cloud service) can be separated.

4 PRELIMINARIES

4.1 Privacy-ABCs

Digital certificates or credentials are essentially a set of attributes signed by a trusted third party, the Credential Issuer. The latter will guarantee that the values in the credential are correct. The most used digital credential is the X.509 certificate (Housley et al., 2002). However, these credentials offer poor privacy properties as a verifier will learn every attribute contained in a credential when the credential's signature is verified. Furthermore, verifiers learn a unique signature and attributes from transactions involving certificates.

Privacy-ABCs, or anonymous credentials, allow for *selective disclosure of attributes*, meaning that a credential owner can hide the attributes that she is not required to disclose to a verifier. Moreover, *multiple transactions with the same credential can be made unlinkable*.

The standards for Privacy-ABC systems has been set by the ABC4Trust project (Sabouri et al., 2012), in which the structure of these credentials and their associated protocols have been specified. The most well known and practical privacy-ABC systems are U-Prove (Paquin and Zaverucha, 2011) and Idemix (Camenisch and Lysyanskaya, 2003; Camenisch and Van Herreweghen, 2002). Idemix is used in our system.

A privacy-ABC owner can be deterred from sharing credentials by including sensitive information in the credential[1] and by linking multiple credentials to the same secret. The person with whom someone shares a credential will learn the sensitive information and will be able to use all her other credentials as well.

An important concept in PACCo are *provable pseudonyms*. Privacy-ABC owners can generate seemingly random pseudonyms of which can be proven in zero knowledge that it was generated with a specific credential (Camenisch et al., 1997; Camenisch and Van Herreweghen, 2002).

4.2 uCentive

The uCentive (Milutinovic et al., 2015) system provides an efficient way for an entity to anonymously bind information to a privacy-ABC. A different party is later able to verify the validity of the data together with the fact that this data belongs to the entity with which it is currently interacting. This is done by combining partially-blinded signatures (PBS) (Abe and Okamoto, 2000) and provable pseudonyms. The blinded part of the PBS contains a provable pseudonym, which is only known to the user. The clear part consist out of data which can be verified by the signer (Figure 1). Once signed,

[1]Note that these attributes can be hidden in every transaction.

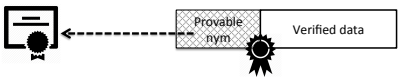

Figure 1: uCentive token. The provable pseudonym is blinded when the signature is made, while other information is not blinded and can be verified.

the user can unblind her provable pseudonym, the result of which forms a *uCentive token*. Later, she can show the signed token to another entity together with a zero-knowledge proof of the fact that she owns the pseudonym embedded in the token.

5 PACCo POLICY LANGUAGE

5.1 Concepts

First, the basic elements that make up a context requirement are explained. Although this paper does not focus on the specific types of context that should be used in access control, an example set of these elements is shown in Table 1.

Context Source (CS): The Context Source identifies from which kind of device the context originates (e.g. GPS, Cell Tower, Wifi, ...).

Context Type (CT): A Context Type is a collection of Context Sources. For example, the context type *location* contains the sources *GPS*, *Cell Tower* and *Wifi*. It is assumed that a mapping $CT \rightarrow [CS_1 ... CS_n]$ is defined and known to the system.

Operation (OP): Context types support one or more operations. For example, the location type supports the operation *inCity*, while the time type supports *isAfter*. Operations are specified in a function format (i.e. with arguments between parentheses). Furthermore, context sources inherit the operations of their parent type (e.g. GPS supports similar operations as the Location type).

Argument (A): Some operations require one or more arguments to be specified. For example, the inCity operation requires an *Accuracy* argument.

Value (V): A value is used in the evaluation of context requirements. Values are matched to context originating from a CS to which an operation has been applied to. Multiple values are denoted using a comma-separator (V_1, V_2, ..., V_n).

The policy language can be extended by defining new context sources or context types. Note, however, that defining a new CT may include new operations, and, hence, arguments. This extensibility is important as future IoT devices might offer interesting opportunities regarding context-aware access control.

Freshness Attribute: Freshness of context is of the utmost importance. This attribute, specified in seconds, indicates how old contextual information may be, i.e. a freshness parameter of '3600' indicates that the context should have been collected at most one hour ago in order to be considered as valid.

Security Attribute: Using context for access-control is not a trivial task as a malicious user could alter and construct information from context sources, or even spoof them. However, context sources can offer varying degrees of security guarantees. Therefore, three security-levels for context sources are defined.

Unchecked information can have been forged, shared as well as spoofed. E.g. originating from a "dumb" device like a simple sensor.

Certified information is *timestamped and signed*. Hence it cannot be spoofed or forged. It can, however, be shared. E.g. originating from an IoT device controlled by a trusted entity.

Verified information can be *verified at the context source* by a third party. Not only the authenticity, but also the ownership of the contextual information is verifiable. Hence, it cannot be shared. E.g. originating from an IoT device that can link context to a user.

The security attribute defines the minimal level of security for a context source (unchecked < certified < verified). GPS, Wifi, NFC and Paired Bluetooth are examples of *unchecked* context sources, as we assume that the sensors providing this information do not offer any authenticity or integrity guarantees. A Cell Tower is assumed to be an *certified* context source[2].

Finally, Time is a *verified* context source as it can be universally verified. IoT devices could also act as context sources. The context type and their associated security parameter which they represent will be determined based on their capabilities. Therefore, IoT devices will provide the services of most certified and verified context sources.

[2]Although smartphone applications do not have access to cell tower authenticity information, such a feature is technically possible and the source is controlled by a trustworthy entity.

Table 1: Examples of context types and sources with their associated operations and arguments.

Context Type / Source	Operation	Argument	Value
Location	inCity	Accuracy	City Name
GPS, Wifi	withinMeters	Distance,Accuracy	Coordinates
Cell Tower	inPolygon	Accuracy	PolygonCoordinates
Time	isBefore	Format	Time (in specified format)
NTP	isAfter	Format	Time (in specified format)
	days	Format	Set of days (in specified format)
Proximity	inRoom	-	Room Identifier
Wifi, NFC	nearDevice	MaxDistance	Device Identifier
Paired Bluetooth			

Context Requirement (CR): A Context Requirement is the combination of a context type, an operation with its arguments and a value. Alternatively, the CT can be replaced by a CS. Furthermore, the CR can optionally define freshness and security attributes. The CR represents a fundamental contextual constraint and has the following form:

$CR := (<CT> \text{ or } <CS>)$
$<OP>(<A_1>, \ldots, <A_n>) <V>$
WITH *freshness='100' security='verified'*

Context Requirement Set (CRS): A requirement set is a conjunction of context requirements. Similar to the context requirement, an CRS can optionally define freshness and security attributes. These attributes are inherited by the CRs that did not specify them.

$CRS := <CR_1> \wedge <CR_2> \wedge \ldots \wedge <CR_n>$
WITH *freshness='100' security='certified'*

Context Policy: Similar to standard authorization systems, the target of a policy is identified by a Subject, Resource and Action. The condition of a policy is *a disjunction of context requirement sets*. It is required to specify a freshness and security attribute in the condition. These attributes are inherited by any CRS that did not specify attributes themselves. If the condition evaluates to 'true', then the policy's effect will be returned (Permit/Deny). When conflicting effects apply for the same target, a conflict resolution strategy such as the assignment of priorities to an effect can be applied. Priorities are (optionally) specified between parentheses.

P: Target{Subject=S,Resource=R,Action=Ac}
Effect{Permit(1)}
Condition
$\{ <CRS_1> \vee <CRS_2> \vee \ldots \vee <CRS_n> \}$
WITH *freshness='100' security='unchecked'*

5.2 PACCo Policy Examples

The first example (Listing 1) shows a PACCo Policy related to the first scenario in Section 3.

Listing 1: "Conference proceedings policy."

```
Proceedings:
Target{Subject='anySubject',
Resource='anyResource',
Action='fileAccess'}
Effect{Permit(1)}
Condition{
(Location inCity(Accuracy='250')
    London
AND Time days(Format='Ddd') Mon,Tue,
    Wed
AND Time isAfter(Format='hh:mm')
    19:00)
(Location inCity(Accuracy='250')
    London
AND Time days(Format='Ddd') Mon,Tue,
    Wed
AND Time isBefore(Format='hh:mm')
    10:00)
(Location inPolygon(Accuracy='0.001')
50.865389:4.672449,50.862388:4.679377,
50.862759:4.674634
AND Time days(Format='Ddd') Mon,Tue,
    Wed
) }
WITH freshness='600'
     security='certified'
```

The target of this policy relates to anybody that wants to access any file on the file-server.

The policy condition has three context requirement sets, the first of which specifies that one's location should be in London (within an accuracy of 250 meters). Furthermore, the time should after 19h00, and the current day should be Monday, Tuesday or Wednesday. The second set is identical except for the fact that the current time should be before 10h00. Finally, the third set also requires the day to be Monday,

Tuesday or Wednesday, while the location should be within a defined area (the conference venue).

The freshness attribute for the condition is set to 600 seconds, and the security attribute is set to *certified*. This will exclude GPS as context source, as it provides unverified context. Cell Tower information is assumed to be certified and it is accurate enough to provide context for the first requirement set. However, it is not accurate enough for the second CRS. Hence, a location service provided by an IoT device at the conference proceedings is required.

The second example (Listing 2) shows a policy related to the second scenario of Section 3.

It targets the external consultant that wants to access project resources. The policy condition requires that the meeting currently takes place and that the external consultant is in the meeting room. Because the proximity requirement has the unchecked security attribute, any source present in the meeting room could be used. Furthermore, the freshness is set to 100 seconds to prevent the consultant from accessing the files once the meeting is over. As an alternative, the consultant is allowed access if she is in the vicinity of the project leader. Note that the condition's security attribute is *verified*, meaning that it should not be possible to share or forge this context. Therefore, the project leader should wear an IoT device that acts as a proximity context source.

Listing 2: "External consultant policy."

```
ProjectResources:
Target{Subject='consultant',
Resource='projectResource',
Action='anyAction'}
Effect{Permit(2)}
Condition{
(  Time days(Format='ddd') Mon
AND Time isAfter(Format='hh:mm')
  10:00
AND Time isBefore(Format='hh:mm')
  12:00
AND Proximity inRoom()
  meeting_room_A WITH security='
  unchecked')
(  Proximity nearDevice()
  ProjectLeaderDev )
}  WITH freshness='100'
        security='verified'
```

6 PACCo

6.1 Overview and Threat Model

Four main entities are present in PACCo: the User, the Service Provider (SP), the Context Verifier (CV) and the different types of Context Provider (CP).

Users want to make use of the services offered by the service provider. They own a smartphone or another smart-device that is capable of gathering context and authenticating with privacy-ABCs. This device runs a PACCo service that is responsible for managing the user's context.

The *Service Provider* offers an online service to its users. However, in order to improve the strength of its current access-control mechanisms, the SP wants to support context-augmented authentication and context-aware authorization. The SP enforces a set of PACCo policies.

The *Context Verifier* is a third party which will verify the context of the user, after which it gives the user a token that the latter can use to authenticate towards the SP. Separating context verification from the SP not only makes it easier for existing SPs to support PACCo, but also gives this system better privacy properties. In addition, the CV manages the trust in different context providers, keeping track of which context providers are trusted, and which are not.

Context Providers are considered as *black-boxes with limited computational capacity*. CPs, however, do have different capabilities. PACCo identifies four types of Context Providers, based on the security guarantees they can provide (cfr. Section 5.1).

- *Unchecked* CPs (e.g. smartphone sensors) simply return unchecked context data.
- *Certified* CPs also return a signature in which the contextual data, the identifier of the context source, and a timestamp is signed (e.g. cell tower or IoT device).
- *Personal Verifiable* CPs collect contextual information about the users themselves (e.g. IoT activity monitor, proximity sensors). Hence, this information is of a privacy-sensitive nature. Personal context sources offer interfaces to both the user and the CV. Users can request it to gather contextual data. The CP will then collect the requested context and wait for the CV to request it. This is considered a *verified* context source as the CV can verify the authenticity of the context and that this belongs to the user.
- *Global* CPs offer contextual information which can be accessed and verified by anyone (e.g. NTP server). As the CV can gather the (global) context itself, this CP is a *verified* context source

The threat model for this system is organized according to involved entity:

Malicious User. A malicious or compromised user will try to gain access to the service without sat-

isfying the context requirements set by the SP. In order to do this, malicious users can try to share contextual information, steal it, or forge it.

Curious SP. A curious SP will not actively try to beat the system. For example, it will not collude with any third party, the CV or CPs. However, it will try to learn as much as possible from the interactions it has with its users.

Compromized SP. A compromised SP could ignore the access control decisions and allow or deny access to users at will.

Curious but Honest CV. The CV will not collude with a third party nor will it deny verifying or certifying context from users. Similar to a PKI, the SP will trust this entity to correctly issue certified context. However, the CV will try to identify users based on its transactions so it can build detailed, context-rich profiles of individual users.

Trusted CPs. A CP is considered to be trusted. It will always provide the correct context to users and CVs. Furthermore, if a CP is able to identify a user, it will *not reveal this information to the CV*.

PACCo assumes that each entity has a copy of the certificates of each CP, CV an SP. These certificates are used to set up secure, authenticated channels to prevent man-in-the-middle and network sniffing attacks. Furthermore, network-based attacks on user-anonymity are not considered. However, a brief discussion is given in Section 7.2.

6.2 PACCo Protocol

Registration. Before users can use the system, they need to obtain a privacy-ABC, the *PACCo-credential*. This is a one-time step, and can be done at the CV (or at an independent credential issuer that the CV trusts). This credential has at least one attribute, a random value known only to the credential owner, called the PACCo-secret. Furthermore, to prevent users from sharing their credentials, standard privacy-ABC strategies can be applied (cfr. Section 4.1).

Users might be required to identify and authenticate themselves towards the CV or credential issuer. However, this does not affect the user's privacy because the issuance of a privacy-ABC is not linkable to its usage (Camenisch and Van Herreweghen, 2002).

Access Request. The remainder of the protocol is shown in Figure 2. To access a service, a user first makes a request, in which she authenticates to the SP (1). If the user already knows the context-requirements, and if she can satisfy them, she can proceed to *step 7*. Otherwise, she will receive the contextual constraints in an access policy (2), and a temporary session token.

Context Collection. The PACCo service running on the user's device is responsible to collect context from *certified* and *unchecked* context providers (3). The context that needs to be gathered is determined by analyzing the access policy. The PACCo service *adds a timestamp and an identifier of the context provider to the information originating from unchecked providers*, after which this data is signed by the service (4).

Context Validation. The context validation (step 5 and 6) is shown in detail in Figure 3.

PACCo allows CVs to validate personal context by accessing the CP directly. Before a user contacts the CV, it will construct a provable pseudonym (*nym1*) using her credential (cfr. Section 4.1). The CP responds with a challenge (*c1*) in which a timestamp is encoded. This is used to create a zero-knowledge proof of ownership of *nym1* (*p1*), which is sent to CP. Due to the computational limitation of a CP, *p1* is not (immediately) verified. Instead, CP stores *p1* and *nym1*.

Next, the user will contact the CV in order to validate her context (5). This request contains the following:

1. A collection of contextual requirements, as defined in Section 5.1.
2. For each unverified or certified requirement, the contextual information and its signature (*data*). This information contains the actual data, the context source and a timestamp.
3. An identifier for each personal verified context source that should be accessed (*CP-ID*).
4. A provable pseudonym, based on the PACCo-credential (*nym1*).

Next, it validates the *verifiable* requirements. Global context, like the current time, is requested from the CP directly, without a risk to user-privacy. To validate context from a personal verifiable provider, the CV contacts the CP and request *p1* using *nym1*. Now, the CV gathered all the contextual data and it can verify whether the context satisfies the requirements, after which it validates *p1*.

Using a new challenge (*c2*), the CV asks the user to create a new proof (*p1'*) for *nym1* in order to be certain that this user is the one that made proof *p1* (i.e. to be certain that the collected context belongs to this user). Next, the user and CV execute the

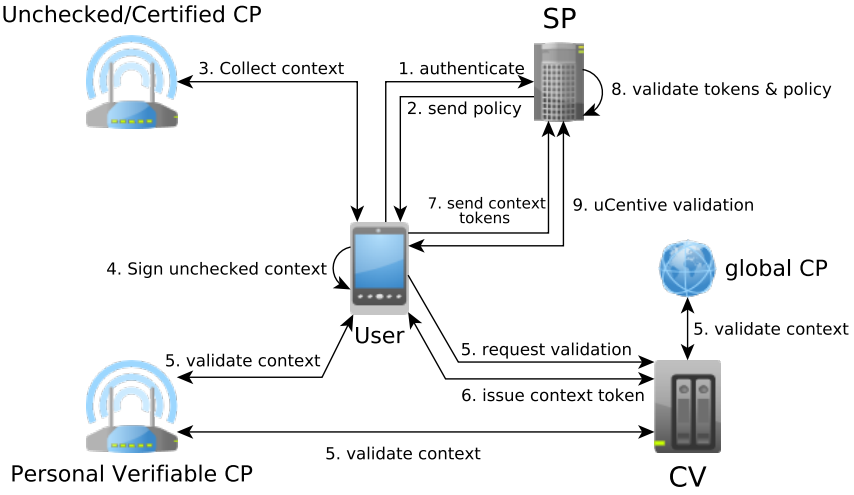

Figure 2: The high-level PACCo protocol.

uCentive protocol in which a partially blinded signature is created on the validated context requirements, the timestamp of context collection, and on a new provable pseudonym (*nym2*). CV will not learn this pseudonym, as it is embedded in the blinded part of the signature, however, it will see the context requirements[3].

Note that CV does not actually sign the contextual data itself, it only signs the context-requirement. This makes a large impact on the amount of information that the SP will learn. At the end of context verification, the user unblinds the received uCentive token.

Contextual Authentication. Now, the user sends a new access request to the SP (step 7 in Figure 2). She shows her session token, after which she shows the uCentive tokens to the SP. The SP will first validate whether the contextual requirements from the access policy are met, after which the tokens themselves are verified (8). Next, the user proves that she and only she owns those tokens by proving in zero-knowledge that the pseudonyms in these tokens are hers (9). Finally, the user is granted access if the information in the tokens can satisfy the context constraints, if the tokens are valid and if they belong to the user.

[3]Details about the cryptographic protocol of uCentive, and how uCentive prevents users from providing someone else's pseudonym can be found in (Milutinovic et al., 2015)

7 DISCUSSION

7.1 Security

Four attack vectors are identified that can be used to break the system's guarantees. Users could try to forge uCentive tokens, share contextual information or try to perform a redirection attack with a personal verified CP. Furthermore, the SP can be attacked directly.

uCentive Token. It is mathematically hard to forge a uCentive token. Furthermore, once such a token is issued, it is linked to the PACCo-credential through a provable pseudonym; it cannot be used by anyone else than the owner of the correct credential.

However, a problem arises for users that are not deterred by the standard privacy-ABC sharing prevention techniques, such as embedding sensitive information in credential attributes. One technique that can be employed is to store the credential's secret using hardware-backed storage in a way that only the PACCo service on the smartphone can access it.

Context from a CP. Malicious or compromised users are able to share certified or unchecked information after it has been collected because there exists no verifiable link between a user and the collected context. Furthermore, users can even forge or spoof unchecked context before the PACCo-service signs it.

However, the way to solve this would require unrealistic modifications from existing infrastructure. PACCo does rely on future IoT devices to offer the

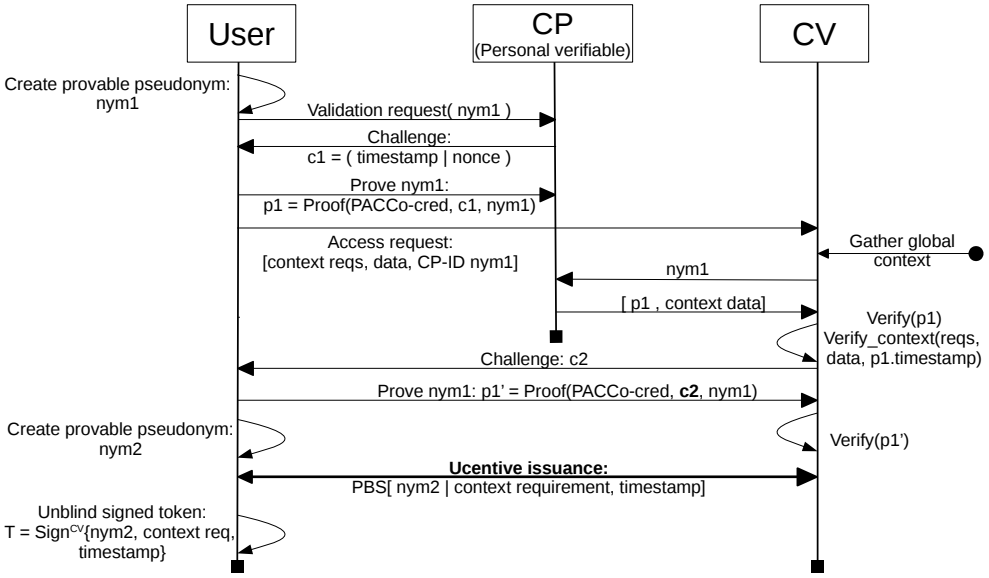

Figure 3: Context verification protocol (see step 6 in Figure 2). Note that all interactions with the CP are only executed if the system needs to validate personal verifiable context.

desired security guarantees. However, in order to allow existing infrastructure and devices with very limited resources to be used, PACCo classifies these devices based on the security guarantees they can offer. *Certified* sources should only be used by an SP if it determines that the probability and impact of context sharing is low. A similar analysis should be done for *unchecked* context sources. *Although the usage of these context sources will never provide an airtight security solution, it will demand more effort from attackers, which might be a sufficient deterrent.*

Request for Verified Context to a CP. Here, two malicious users could execute a redirection attack, where one user will relay the pseudonym, challenge and proof. Normal man-in-the-middle attacks are assumed to be mitigated by setting up secure connections using locally stored certificates. However, it is still possible if the end-user and 'middle-user' work together. This is, however, not an easy attack to execute. Furthermore, the practicality of this attack can be further diminished by distance bounding techniques (Singelee and Preneel, 2005; Brands and Chaum, 1993).

Compromized SP. Huge problems occur when the entity responsible for enforcing access control decisions is compromized. This can be identified by frequently auditing the system and its logs. PACCo has an added bonus for this, as the proofs made in the uCentive protocols can be verified by any third party at any time. In addition, these proofs are unforgeable, making hiding malicious activity a lot harder.

7.2 Privacy

Unlinkable Transactions with the CV. The only information that could identify a user to the CV is the contextual data which it needs to validate. This information might in some cases be enough to limit the user's anonymity set with regard to the CV to a few possibilities (e.g. inhabitants of a home). *This is not the case for many other use-cases* (e.g. allow access to information to students on campus). Furthermore, one validation of several contextual requirements that together can identify the user, can be split in multiple, unlinkable validations. Here, a trade-off between privacy and performance should be made.

Network based timing attacks are also not reliable if a large enough set of users access the SP and CV at the same time. Furthermore, the interactions of the user between SP and CV are completely separate. This allows the user to wait a random amount of time between contacting the SP and CV (within freshness limits) in order to make this kind of tracking more difficult. A PACCo service that periodically collects and verifies context in the background would be able to implement these strategies without loss of user experience.

Other network-based attacks, such as IP-tracking could still be used. However, IP-tracking is not always reliable as the PACCo service is running on a mobile phone which has a more frequently changing IP address than a computer. The service could also

connect to the CV through an anonymous network.

Other information that a CV could learn originates from the Idemix proof and the issuance of the uCentive token. However, multiple Idemix transactions cannot be linked to each other. In addition, the issuance of uCentive tokens is not linkable to the spending thereof (Milutinovic et al., 2015).

Finally, if users are required to identify themselves to a CP, PACCo assumes that these CPs will not share these identities with the CV. However, PACCo allows users to choose the CPs that are accessed. Therefore, they should only use CPs that are managed by a trusted organization. Note that in scenario 2 of Section 3, the company deploys CPs and the CV. Here, the anonymity of users at the CV could be lost, which is acceptable in a company setting. However, the privacy benefits related to the SP, which is external to the company, remain intact.

Minimal Information Learning by the SP. The SP learns only a small amount from a transaction. It learns 1) that the user satisfies a set of context constraints and 2) the times when the context is collected; the SP does not learn the precise context of the user.

However, whether the SP will be able to identify a user will mostly be determined by the standard authentication mechanism of the SP. In the optimal case, an anonymous authentication system like Idemix, is used, which causes the SP to only learn that the user has a right to access the service, and that she satisfies the context constraints. The SP will be able to link contextual information to specific users in the likely case that it uses a standard authentication mechanism. However, this information is often obvious (e.g. the user is on campus when accessing student material).

Finally, the timestamp in the context requirements should not be unique. Here, the CV can offer a solution by decreasing the timestamp's granularity.

7.3 Performance

Setup Our prototype consists out a PACCo service running on a mobile phone, two server components: a CV and an SP, and three CPs: an NTP server, an NFC chip and a second phone acting as a *personal verifiable* proximity CP. The smartphones are both Samsung Galaxy S3 with a 1.4 GHz processor. A workstation with an Intel Core i7-3770 CPU hosts the CV and SP. The prototype is implemented in Java and uses the PriMan framework (Put et al., 2014) to access Idemix and uCentive functionality.

The client owns one PACCo-credential with one attribute, the PACCo-secret (2048 bit modulo and a 1632 bit commitment group modulo). The PACCo service uses 2048 bit RSA signatures in combination with SHA-256. The partially blinded signatures have 1024 bits modulo and 160 bit generator groups[4]. For each test, 100 samples were taken of which the mean value and standard deviation is shown. The core measurements of our tests do not take network time and the time it takes to obtain contextual information (i.e. the time it takes for the sensors to measure the data) into account. As such, we aim to provide a clear view of the overhead that PACCo's security and privacy measures introduce.

7.4 Test Case

Our test case will consider the second scenario explained in Section 3, as it includes the verification of personal verified context. The policy for this scenario is shown in Listing 2. In order to satisfy this policy, one needs to satisfy one of the following two constraints: the current time should be between 10h00 and 12h00 and the user should be in a specific meeting room, or the user should be in the proximity of the project leader. The security requirement for the proximity context source in the meeting room can be 'unchecked', while the other sources are 'verified'.

The NTP server is a *global* context provider; the CV can gather and verify its context. The NFC chip is a *unchecked* context provider, as it is scanned using the client's smartphone. The data will be signed by the PACCo service running on the phone. The second phone provides *personal verifiable* context. It will take act the role of a CP in the protocol shown in Figure 3. Table 2 shows the results.

Unchecked Context. Unchecked context is collected through the smartphone sensors after which the PACCo service performs an RSA signature. This operation takes 5 ms on our phone, while the workstation requires 2 ms in order to verify this signature.

Certified context is handled similar to unchecked context. However, the PACCo service will not create a digital signature because one is already provided by the CP. Hence, the CV only needs to verify signatures for certified context.

Verifiable Context. Collecting *personal* verifiable context is one of the computationally expensive parts of PACCo. Creating a zero knowledge proof for one provable nym takes 232 ms on our phone, while the workstation requires almost 80 ms to verify it. Note

[4]Note that, due to the limited validity of these signatures, the security parameters can be relaxed compared to certificate signatures.

Table 2: Performance numbers for PACCo operations in ms, standard deviation is listed between parentheses.

Operation	Client	CV	SP
Unverified context			
RSA Signature	5 (1)	2 (0)	-
Verifiable context			
Prov. nym	232 (9)	79 (3)	-
uCent. gen.			
Earn uCent	226 (8)	46 (2)	-
Authentication			
Policy verif.	-	-	2 (0)
Spend 1 uCent	225 (12)	-	84 (6)
Spend 5 uCents	255 (12)	-	91 (5)
Total			
a. NTP & NFC	456 (15)	48 (2)	86 (6)
b. Smartphone	915 (20)	204 (4)	86 (6)

that these operations have to be executed twice: one proof is sent to the CP and the second is used to prove that the first one is made by the same person. The CV verifies both proofs.

Collecting *global* verifiable context such as time from an NTP server, is performed by the CV. As explained above, this is not included in our performance measurements as the amount of time it takes to verify is not affected by PACCo.

Generation of uCentive Token. Once the context has been validated, the uCentive protocol is executed, in which a partially blinded signature is created. The creation of one uCentive signature requires 226 ms from the client while the server requires 46 ms. Furthermore, this process scales linearly with the amount of signatures.

Authentication. The user will sends uCentive tokens to the SP. Next, the SP validates whether the contextual information in these tokens satisfy the the policy. This is a relatively inexpensive step, as it requires on average less than 2 milliseconds.

Furthermore, the SP validates the tokens. The client and SP execute the protocol to spend uCentive tokens, in which the client proofs that the pseudonyms in the tokens belong to her. Our smartphone requires on average 225 ms, while the CV, running on our workstation, needs 84 ms to verify.

Spending and verifying additional uCentive tokens is cheap. Spending 5 tokens requires 255 ms on the client and 91 ms on the server. This is interesting as users can spend a set of tokens at once that were collected over time.

The Big Picture. As shown in Table 2, PACCo needs under half a second on a smartphone if no personal verifiable context is required (a. NTP & NFC), while a CV requires under 50 ms. The validation of a personal verifiable requirement (b. Smartphone) will add about half a second to the client's time and near 150 ms to the CV's time. The SP spends the majority of time verifying a uCentive token. Note that this time increases only little with additional tokens.

Disregarding network time and the time for the CPs to return contextual information, the whole PACCo protocol takes just over half a second to verify the first requirement set (a.). The verification of information from a personal verifiable context source (b.) takes just over 1.2 seconds.

8 CONCLUSION

This paper presented PACCo, a privacy-friendly access control system with context. We have shown with our policy language how to represent contextual requirements. Furthermore, we have designed and implemented our system, which focuses on the secure and privacy-friendly validation of context. In addition, our experimental evaluation shows that our system produces an acceptable overhead which can range from about 600 ms to 1.2 seconds.

However, future work should address not only the security of context, but also the quality of context without sacrificing privacy. Exactly which context should be used in access control and what techniques can be used to improve the decision making are interesting questions, especially with the opportunities provided to us by the Internet of Things. Furthermore, user experience and factors, like power-usage and background-scheduling should be optimized.

Finally, the techniques explored in this paper could also implemented in a distributed access control system where evidence can be gathered by different systems (verifiers). A service provider can then combine different pieces of evidence while being able to verify whether all these pieces belongs to the same user.

REFERENCES

Abe, M. and Okamoto, T. (2000). Provably secure partially blind signatures. In *Advances in CryptologyCRYPTO 2000*, pages 271–286. Springer.

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.

Adams, A. and Sasse, M. A. (1999). Users are not the enemy. *Communications of the ACM*, 42(12):40–46.

Ardagna, C. A., Di Vimercati, S. D. C., Foresti, S., Grandison, T. W., Jajodia, S., and Samarati, P. (2010). Access control for smarter healthcare using policy spaces. *Computers & Security*, 29(8):848–858.

Atluri, V. and Chun, S. A. (2007). A geotemporal role-based authorisation system. *International Journal of Information and Computer Security*, 1(1-2):143–168.

Bhatti, R., Bertino, E., and Ghafoor, A. (2005). A trust-based context-aware access control model for web-services. *Distributed and Parallel Databases*, 18(1):83–105.

Brands, S. and Chaum, D. (1993). Distance-bounding protocols. In *Advances in CryptologyEUROCRYPT93*, pages 344–359. Springer.

Camenisch, J. and Lysyanskaya, A. (2003). A signature scheme with efficient protocols. In *Security in communication networks*, pages 268–289. Springer.

Camenisch, J., Stadler, M., Camenisch, J., and Camenisch, J. (1997). *Proof systems for general statements about discrete logarithms*. Citeseer.

Camenisch, J. and Van Herreweghen, E. (2002). Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM.

Groopman, J. (2015). Consumer perceptions of privacy in the internet of things. *Altimeter Group*.

Hayashi, E., Das, S., Amini, S., Hong, J., and Oakley, I. (2013). Casa: Context-aware scalable authentication. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS '13, pages 3:1–3:10, New York, NY, USA. ACM.

Hintze, D., Findling, R. D., Muaaz, M., Koch, E., and Mayrhofer, R. (2015). Cormorant: Towards continuous risk-aware multi-modal cross-device authentication. UbiComp/ISWC'15 Adjunct.

Housley, R., Polk, W., Ford, W., and Solo, D. (2002). Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile.

Hu, J. and Weaver, A. C. (2004). A dynamic, context-aware security infrastructure for distributed healthcare applications. In *Proceedings of the first workshop on pervasive privacy security, privacy, and trust*, pages 1–8. Citeseer.

Hulsebosch, R., Salden, A., Bargh, M., Ebben, P., and Reitsma, J. (2005). Context sensitive access control. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 111–119. ACM.

Jafarian, J. H. and Amini, M. (2015). Camac: A context-aware mandatory access control model. *The ISC International Journal of Information Security*, 1(1).

Jin, X., Krishnan, R., and Sandhu, R. S. (2012). A unified attribute-based access control model covering dac, mac and rbac. *DBSec*, 12:41–55.

Kulkarni, D. and Tripathi, A. (2008). Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 113–122. ACM.

Matheus, A. and Herrmann, J. (2008). Geospatial extensible access control markup language (geoxacml). *Open Geospatial Consortium Inc. OGC*.

Milutinovic, M., Dacosta, I., Put, A., and Decker, B. D. (2015). ucentive: An efficient, anonymous and unlinkable incentives scheme. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 588–595. IEEE.

Paquin, C. and Zaverucha, G. (2011). U-prove cryptographic specification v1. 1. Technical report, Microsoft Technical Report, http://connect. microsoft. com/site1188.

Put, A., Dacosta, I., Milutinovic, M., and De Decker, B. (2014). Priman: Facilitating the development of secure and privacy-preserving applications. In *SEC*, pages 403–416. Springer.

Ray, I. and Toahchoodee, M. (2007). A spatio-temporal role-based access control model. In *Data and Applications Security XXI*, pages 211–226. Springer.

Rissanen, E. et al. (2013). extensible access control markup language (xacml) version 3.0.

Riva, O., Qin, C., Strauss, K., and Lymberopoulos, D. (2012). Progressive authentication: Deciding when to authenticate on mobile phones. In *USENIX Security*, pages 301–316.

Sabouri, A., Krontiris, I., and Rannenberg, K. (2012). *Attribute-based credentials for Trust (ABC4Trust)*. Springer.

Shebaro, B., Oluwatimi, O., and Bertino, E. (2015). Context-based access control systems for mobile devices. *Dependable and Secure Computing, IEEE Transactions on*, 12(2):150–163.

Singelee, D. and Preneel, B. (2005). Location verification using secure distance bounding protocols. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 7–pp. IEEE.

Vimercati, S. D. C. D., Foresti, S., Jajodia, S., Paraboschi, S., Psaila, G., and Samarati, P. (2012). Integrating trust management and access control in data-intensive web applications. *ACM Transactions on the Web (TWEB)*, 6(2):6.

Yuan, E. and Tong, J. (2005). Attributed based access control (abac) for web services. In *2005 IEEE International Conference on Web Services*. IEEE.