# AppPETs: A Framework for Privacy-Preserving Apps

Erik Sy
University of Hamburg
Hamburg, Germany
sy@informatik.uni-hamburg.de

Tobias Mueller
University of Hamburg
Hamburg, Germany

Matthias Marx
University of Hamburg
Hamburg, Germany

Dominik Herrmann
University of Bamberg
Bamberg, Germany

## ABSTRACT

The protection of users' privacy is an important property of smartphone apps. However, most apps do not honour the privacy of their users. One reason for this deficiency is that developers are unaware of and struggle with the implementation of protection techniques. In this paper, we propose an infrastructure to simplify the use of privacy enhancing technologies (PET) in apps. Furthermore, AppPETs provides incentives for app developers to build privacy-preserving apps. The AppPETs concept consists of a client-side library (P-Lib), a hosting environment for privacy services (P-Services), an anonymous communication network, and an audit process. AppPETs supports various types of anonymity that are relevant for real-world use cases. It closes the gap between research and practice in this field and will foster the adoption of PETs on mobile devices.

## CCS CONCEPTS

• **Security and privacy** → *Pseudonymity, anonymity and untraceability*; *Privacy-preserving protocols*; *Mobile and wireless security*; Privacy protections;

## KEYWORDS

Privacy Enhancing Technologies; Anonymity; Mobile devices; Security; Framework

**ACM Reference Format:**
Erik Sy, Tobias Mueller, Matthias Marx, and Dominik Herrmann. 2018. AppPETs: A Framework for Privacy-Preserving Apps. In *SAC 2018: SAC 2018: Symposium on Applied Computing , April 9–13, 2018, Pau, France.* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.1145/3167132.3167415

## 1 PROBLEM DESCRIPTION

Even though there exist many technical measures to address data protection on mobile devices [5, 6, 11], app developers are mostly not aware of them [1]. As a consequence, their apps are not designed to be in line with the data protection goals [4]. Moreover, research work showed that most software developers have difficulties in differentiating between security and privacy [12]. Balebako et al. [1] suggest to design tools to facilitate the implementation of privacy best practices for app developers. Following this suggestion, we provide a framework that simplifies the usage of PETs for developers.

Results reported by Sheth et al. [17] indicate that **anonymisation techniques are well perceived by users and app developers as a means to reduce privacy concerns**. For this reason, we designed AppPETs in such a way that six types of anonymity can be protected with it.

While app developers perceive anonymisation techniques as effective to protect users' privacy, there exist hardly any tools to assist with the implementation of these methods. One of the few examples is the Sharemind SDK [5]. However, it addresses a very specific use case (end-to-end encrypted data processing) and is subject to practical limitations because it relies on secure multi-party computation to protect the data content against a computing server [5].

More practical are frameworks that facilitate the inclusion of ACNs into mobile apps, for instance Onionkit [11] and CPAProxy [6]. However, various types of ACNs (e. g., proxies, VPNs, mix networks) differ in terms of the underlying attacker model and performance, which makes the choice for an adequate anonymization method a challenging task. The AppPETs concept reduces this complexity by **providing multiple anonymisation techniques through a privacy framework** and helping the app developer to choose the adequate anonymization method for a set of common use cases.

## 2 APPPETS DESIGN

The AppPETs infrastructure, as shown in Fig. 1, consists of a client-side privacy library (P-Lib), an ACN, and Privacy Services (P-Services). The P-Lib provides functionalities for app developers to reduce the burden of creating privacy-preserving apps. This includes methods to use PETs like ACNs, private information retrieval (PIR), and encryption algorithms.

An ACN provides unlinkability between the client and the server on the network layer. The choice of the anonymisation technology used should take the limited resources of mobile devices into account and provide a sufficient performance to support a broad field of apps.

PETs such as PIR [8, 14] or secure multi-party computation [10] require servers to support these protocols and algorithms. P-Services provide such functionalities and therefore make it easier for app developers to use these PETs.

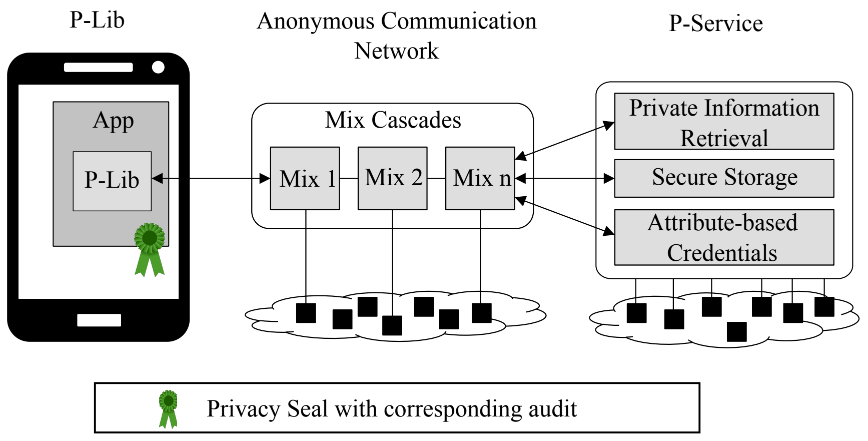
P-Lib
Anonymous Communication Network
P-Service
App
P-Lib
Mix Cascades
Mix 1
Mix 2
Mix n
Private Information Retrieval
Secure Storage
Attribute-based Credentials
Privacy Seal with corresponding audit

**Figure 1: Overview of the AppPETs concept.**

The AppPETs infrastructure is complemented by a privacy seal which is awarded after an audit. The audit validates the compliance of an app with the data protection guidelines of the privacy seal. An app could be advertised with that privacy seal to show its compliance, which in turn attracts users and differentiates the app from its competitors.

## 2.1 P-Lib

The P-Lib is a software library that provides generic functions for the development of privacy-preserving apps. This includes functions to use encryption algorithms, the ACN, privacy-preserving algorithms to access hardware sensors or private data on the mobile device and P-Services such as Attribute-based Credentials (ABCs) [16] and PIR. ABCs describe a mechanism to authenticate certain attributes of an entity, without disclosing additional attributes.

In this context a privacy-preserving algorithm to access a hardware sensor may provide an app with coarse-grained location information like postal codes instead of accurate coordinates from the GPS sensor [13]. An example for privacy-preserving access of data on a mobile device would be a function, which strips off metadata such as GPS coordinates from photos, if that information is not required.

P-Lib provides implementations of PETs to an app developer and therefore reduces the hurdles to make use of PIR, ABC and secure multi-party computation. The P-Lib is aligned with the APIs of the P-Services to support their functionality. It can be used on the one hand to protect the confidentiality of private data against sinks, e. g., by encrypting data before it is transmitted to a remote server. On the other hand it can be used to protect sensitive data before it gets into the app context. This allows developers to access only a subset of the functionality of a protected resource, such as sensors or other resources which are usually protected by the operating system's permission system.

## 2.2 Anonymous Communication Network

An ACN allows an app to communicate anonymously with P-Services or other websites in the Internet. To support a wide range of apps, such as messaging, gaming or streaming apps, the anonymity network should target the different needs of these mobile applications. It should provide high performance in terms of throughput and latency if needed. Lightweight ACNs (e. g., SHALON [15]) with higher bandwidth throughput and a lower round trip time (compared to the popular Tor network [9]) may be needed to ensure good usability on mobile networks. Depending on the anonymity network, P-Services could be designed as hidden services, such that the traffic never leaves the anonymity network.

## 2.3 P-Service

P-Services are the server infrastructure of the AppPETs design. They provide APIs for PETs like PIR, ABC and secure multi-party-computation. They can be accessed via the P-Lib, which allows the app developer to directly include these PETs in their apps.

Besides services for PIR and ABC there are also P-Services for data storage, anonymous payments [19], and privacy-preserving targeted advertisement [18].

## 2.4 Audit

The AppPETs concept includes an optional app audit to validate the data protection behaviour of an app. Apps which successfully pass the audit are certified with a privacy seal. The audit consists of a written data protection statement, static and dynamic program analysis techniques, a verification mechanism, and a privacy seal.

The *data protection* statement defines the requirements which an app needs to satisfy to become certified with the privacy seal. During the audit static and dynamic program analysis is performed on the closed-source app bundles that would be normally downloaded from the app store. This ensures that the app uses the P-Lib and P-Services as intended and that the requirements of the data protection statement are satisfied. Moreover, side-channels that could be used for malicious behaviour can be detected.

The *privacy seal*, which is granted to an app developer after a successful audit, is an incentive for the app developer to use AppPETs and to comply with the requirements. Furthermore, the seal helps enterprises and public authorities to ensure that their employees use only privacy-preserving apps on their mobile devices in order to be compliant with data protection standards.

## 3 IMPLEMENTATION AND EVALUATION

In this section, we first present our attacker model before we elaborate on how the design of AppPETs makes it possible to achieve six different types of anonymity (cf. Sec. 3.2). We then evaluate how AppPETs benefits contemporary mobile applications by investigating the features that are in use in popular apps.

### 3.1 Attacker Model

AppPETs allows developers to build apps which may require different types of anonymity. This anonymity can only be assured within a defined attacker model. We make the following assumptions about adversaries against which AppPETs is supposed to offer protection:

- An adversary cannot break cryptographic primitives.
- An adversary cannot compromise the physical device of an honest user running the app.
- An adversary is limited to the security assumptions of the respective PETs used for the implementation of an app.

AppPETs considers three types of adversaries: First, outsiders (eavesdroppers on the network, other users), secondly app developers (who try to spy on the users of their apps), and thirdly P-Service operators (who try to spy on the users of apps that use their P-Service). The auditor is not assumed to be malicious. However, we have to assume that auditor and malicious app developers do not collude.

### 3.2 Achieving Anonymity

This subsection details how AppPETs helps a developer to create apps that offer the respective types of anonymity.

**Sender anonymity** requires at least an ACN to protect the identity of the uploading device. If an online service restricts uploads to authorised users, then ABCs [2] might be used to anonymously authorise users before they can upload some data to the server. If the adversary can launch a successful attack against the ACN or the ABC system the uploader can be deanonymised.

**Receiver anonymity** can be implemented with an ACN and, if the download is restricted to an authorised audience, ABCs to anonymously authorise the download. Besides the attacks against the ACN or the ABCs (see previous paragraph) an adversary might include malware in the data file which leaks information about the downloading device to a given host. Depending on the content of the data the adversary might also expect a certain behaviour from the downloading entity like visiting a specific website. This context information can be used to reduce the anonymity set.

**Server anonymity** requires the use of an ACN which supports hidden services as Tor [9] does. The adversary can either try to attack the software running on the server directly or make use of limitations of the ACN [3].

**Data file anonymity** can be achieved with a non-deterministic encryption of a data file, where several times of encrypting the same data file results each time in different cipher texts. If the adversary has no access to the decryption key and the used encryption algorithm is secure, then the adversary might try an brute-force attack.

**Query anonymity** requires that the server and client support a PIR scheme [7]. The adversary tries to find out what data was requested by the client. The adversary might try to exploit limitations of the used PIR scheme.

**Data modification anonymity** can be achieved with non-deterministic encryption. The client retrieves a data set from the server in order to hide the changed data within that set. Then it can either modify the data by adding, deleting, or replacing information in the data set or leave the data set unchanged. Afterwards it encrypts the data set with a non-deterministic algorithm and uploads it to the server. The adversary cannot break cryptographic primitives and might only try a brute-force attack, if he has no access to the decryption key or and a secure encryption algorithm was used.

### 3.3 Quantitative Analysis of Popular Apps

In order to determine what kinds of functionality are required in typical apps, we have evaluated 100 popular apps from Apple's App Store in September 2017. To obtain a broad variety of apps, we picked the Top 10 apps from ten randomly chosen categories. We considered the following categories:

We specifically looked for the following kinds of functionality, which can be implemented with AppPETs in a privacy-preserving manner:

**Network** Does the app communicate over the Internet? We determined this property by starting the app and observing the network traffic the device generates. In order to distinguish between app-specific traffic and background traffic of the operating system, we installed a custom X.509 root certificate on an Apple iPhone 6s and used *mitmproxy* to inspect the encrypted TLS traffic.

**User accounts** Does the app offer a way to register an account or log into it? We started each app and browsed the user interface. We searched for widgets that indicate a possibility to register a personal profile or log into the service.

**In-app purchases** Does the app ask for payments to be made from within the app itself? We determined this property by analysing the metadata of each app for the in-app purchases flag.

**Multi-user interaction** Does the app offer some way of communicating with another user of the service? We determined this property by looking at descriptions of the app in the associated metadata and by starting the mobile application and navigating the UI in order to find elements indicating some form of communication, e. g., whether it offers to send a message to another user.

The results of our analysis are shown in Fig. 2, Fig. 3 and Fig. 4. All of the 100 apps require network functionality. User accounts can be used by 73 apps, communication with other users can be performed by 52 apps, and payments can be made with 40 apps. All apps in the social network category have both the user account and multi-user communication functionality. The category with the most in-app purchases is games (9 of 10 apps) followed by Health and Fitness (8 of 10 apps). In principle, each of the investigated apps could be improved with AppPETs in terms of privacy.

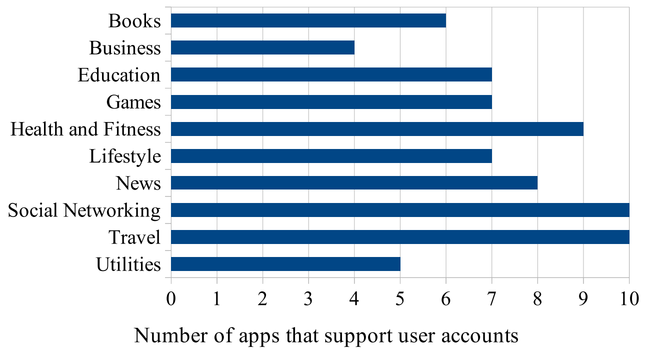

**Figure 2: Number of top 10 gratis apps in iOS App Store Categories that support the feature *user accounts* (in total: 73 %).**

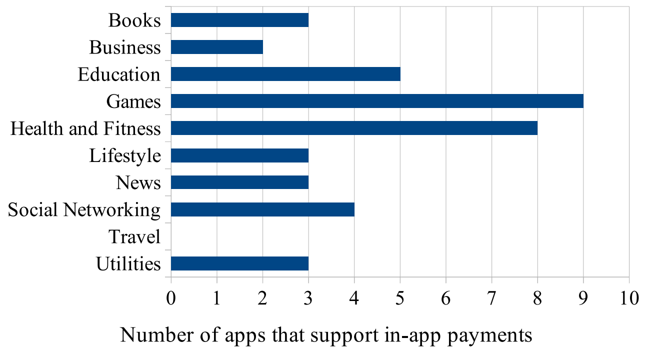

**Figure 3: Number of top 10 gratis apps in iOS App Store Categories that support the feature *in-app purchases* (in total: 40 %).**

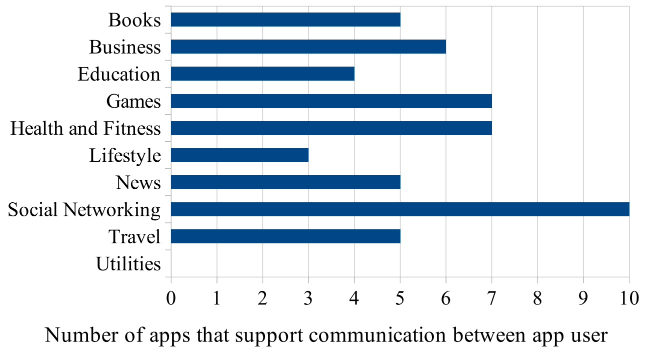

**Figure 4: Number of top 10 gratis apps in iOS App Store Categories that support the feature *communication between app users* (in total: 52 %).**

## 4 CONCLUSION

In this paper we have introduced AppPETs, a framework that strives to make it easier for app developers to access and use privacy enhancing techniques. We performed a quantitative analysis consisting of 100 popular iOS apps in order to determine common features that may cause inadvertent leakage of sensitive user data. The proposed AppPETs framework provides implementations of PETs that allow app developers to create privacy-preserving apps, which are prevented from unlimited spying on their users.

App developers can use this framework to include PETs such as anonymous communication networks (ACN), attribute-based credentials or private information retrieval into their apps. Furthermore, we described six types of anonymity which can be realised with the introduced framework and showed their usage with example apps.

On the long term we hope that AppPETs will be adopted by many app developers and thus contribute to the dissemination of privacy enhancing techniques.

**Acknowledgements.** This work was supported by BWFG/LFF as well as BMBF within the AppPETs and AN.ON-next projects.

## REFERENCES

[1] R. Balebako, A. Marsh, J. Lin, J. I Hong, and L. F. Cranor. 2014. The privacy and security behaviors of smartphone app developers. In *USEC'14*.
[2] J. Bethencourt, A. Sahai, and B. Waters. 2007. Ciphertext-Policy Attribute-Based Encryption. In *P IEEE S SECUR PRIV*. IEEE.
[3] A. Biryukov, I. Pustogarov, and R.-P. Weinmann. 2013. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *P IEEE S SECUR PRIV*. IEEE.
[4] K. Bock, W. Ernestus, M. Kamp, L. Konzelmann, T. Naumann, U. Robra, M. Rost, G. Schulz, J. Stoll, U. Vollmer, and M. Wilms. 2016. The Standard Data Protection Model - A concept for inspection and consultation on the basis of unified protection goals. In *Conference of the Independent Data Protection Authorities*.
[5] D. Bogdanov, S. Laur, and J. Willemson. 2008. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *ESORICS*, Vol. 5283. Springer.
[6] Ursache C.-V. 2017. CPAProxy: A thin Objective-C wrapper around Tor. https://github.com/ursachec/CPAProxy. (2017).
[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. 1995. Private Information Retrieval. In *FOCS*. IEEE Computer Society.
[8] D. Demmler, A. Herzberg, and T. Schneider. 2014. RAID-PIR: Practical Multi-Server PIR. In *CCSW*. ACM.
[9] R. Dingledine, N. Mathewson, and P. F. Syverson. 2004. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*. USENIX.
[10] T. Dugan and X. Zou. 2016. A Survey of Secure Multiparty Computation Protocols for Privacy Preserving Genetic Tests. In *CHASE*. IEEE.
[11] Guardian Project. 2017. OnionKit for Android. https://guardianproject.info/code/onionkit. (2017).
[12] I. Hadar, T. Hasson, O. Ayalon, E. Toch, M. Birnhack, S. Sherman, and A. Balissa. 2017. Privacy by designers: software developers' privacy mindset. *Empirical Software Engineering* (2017).
[13] S. Jain and J. Lindqvist. 2014. Should I Protect You? Understanding Developers' Behavior to Privacy-Preserving APIs. In *USEC'14*.
[14] R. Ostrovsky and W. Skeith. 2007. A Survey of Single-Database Private Information Retrieval: Techniques and Applications. *PKC'07)* (2007).
[15] A. Panchenko, B. Westermann, L. Pimenidis, and C. Andersson. 2009. SHALON: Lightweight Anonymization Based on Open Standards. In *IEEE IC COMP COM NET*. IEEE.
[16] A. Sabouri, I. Krontiris, and K. Rannenberg. 2012. Attribute-based credentials for trust (ABC4Trust). In *TrustBus*. Springer.
[17] S. Sheth, G. E. Kaiser, and W. Maalej. 2014. Us and them: a study of privacy requirements across north america, asia, and europe. In *ICSE*. ACM.
[18] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. 2010. Adnostic: Privacy Preserving Targeted Advertising. In *NDSS*. The Internet Society.
[19] B. Westermann. 2009. Security Analysis of AN.ON's Payment Scheme. In *NordSec' 09*. Springer.