C. Warin, D. Seeger, S. Shams, and D. Reinhardt. PrivXR: A Cross-Platform Privacy-Preserving API and Privacy Panel for Extended Reality. Proceedings of the 22nd International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Work-in-Progress), 2024.

PrivXR: A Cross-Platform Privacy-Preserving API and Privacy Panel for Extended Reality

Chris Warin
Institute of Computer Science
University of Göttingen
Göttingen, Germany
warin@cs.uni-goettingen.de

Dominik Seeger
Institute of Computer Science
University of Göttingen
Göttingen, Germany
dominik.seeger@stud.uni-goettingen.de

Shirin Shams
Institute of Computer Science
University of Göttingen
Göttingen, Germany
shams@cs.uni-goettingen.de

Delphine Reinhardt
Institute of Computer Science
Campus Institute Data Science (CIDAS)
University of Göttingen
Göttingen, Germany
reinhardt@cs.uni-goettingen.de

***Abstract*—*Extended Reality* (XR) technologies are rising in popularity and affordability, while including more sensors for new features at each new generation, thus becoming increasingly more pervasive. While this shapes up interconnected experiences across *Augmented Reality* (AR), *Mixed Reality* (MR), and *Virtual Reality* (VR) devices, it also introduces privacy threats for users, especially related to their sensible biometric data. Despite various propositions for privacy-enhancing technologies tailored to XR in academia, there are still not enough options for end-users to be informed about privacy risks and act upon them. Therefore, we present a work-in-progress solution, consisting of a user-friendly privacy panel, and a cross-platform privacy-preserving *Application Programming Interface* (API). Our solution aims to provide more awareness about potential privacy risks in XR, while enabling users to define access to XR features, better protecting their privacy by modifying the input data. Eventually, we aim for this work to become a viable choice for end-users of current and future generations of XR devices—especially in the context of cross-platform, multi-user experiences, which are expected to become the norm.**

***Index Terms*—Extended Reality, Privacy-Enhancing Technologies, API**

I. Introduction

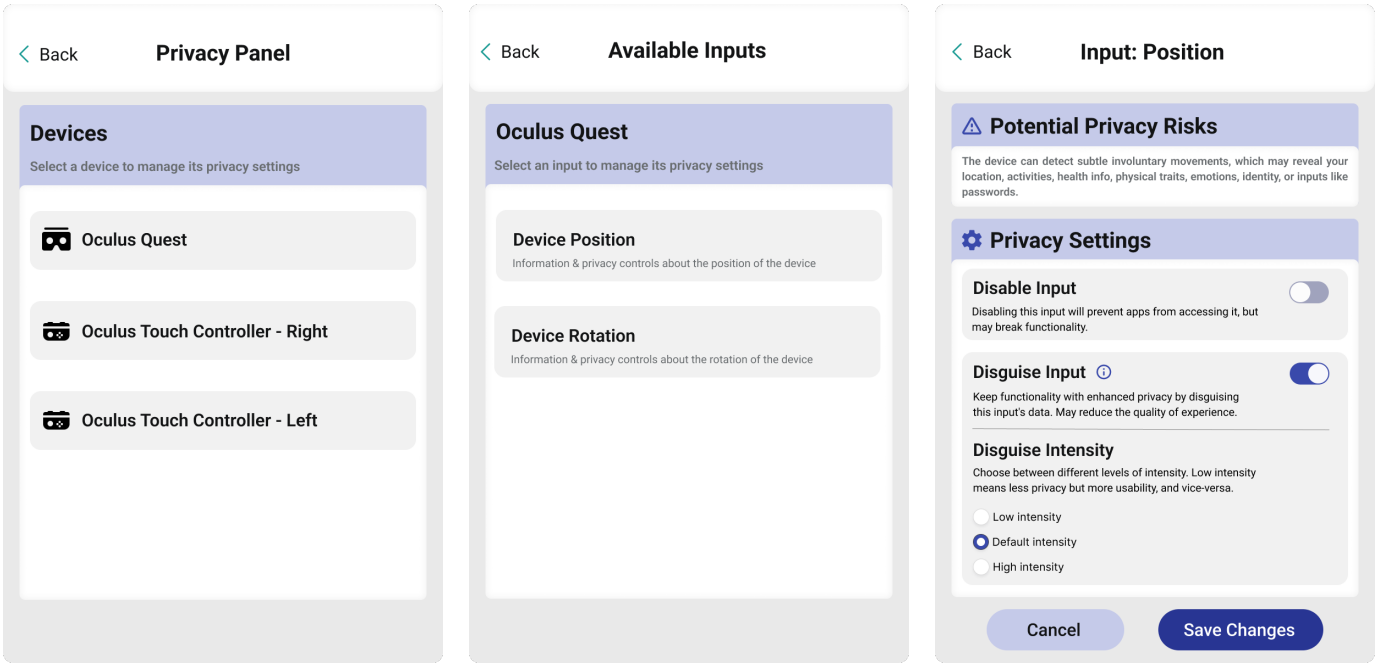
XR is becoming more mainstream and more affordable [1]. However, large amounts of data are collected from users and their environments in order for the technologies to work, which may endanger their privacy [2], and the privacy of bystanders [3]. In particular, recent works show the sensitivity of biometric data used in XR devices (e.g., user identification based on body movement data [4], [5], data inference from eye tracking data, such as intentions or personal preferences [6]). Such sensible data pose threats to the privacy of XR users, when in the hands of malicious third-party developers, or global companies who may collect these data for, e.g., targeted advertising ends. Therefore, in the current state-of-the-art, the sensible but mandatory nature of XR data compels its users to sacrifice privacy in order to use these new technologies and take part, e.g., in the upcoming Metaverse.

The XR research community has proposed various solutions to enhance user privacy in AR and VR systems, e.g., [3], [7]–[11]. Recently, [9] have proposed an incognito mode for VR through a user-friendly privacy panel that enforces privacy on sensor data through differential privacy algorithms [9] (a preliminary improvement with deep-learning models is also proposed in [10]). Their approach require end-users to patch every third-party application they want privacy control over, which then allows them to access the privacy panel in the patched application. While this approach is very versatile, user adoption is questionable because of the required technical knowledge and the incurred overhead for users. Furthermore, other tracks remain to be explored, such as usability evaluations, and cross-platform implementations to match the Metaverse's multi-user and multi-platform nature [12], [13].

Therefore, there is still a need for user-friendly privacy-enhancing technologies in consumer devices, especially with respect to biometric data. Such solutions are needed in order to better protect privacy, raise XR users' privacy awareness, and, eventually, foster user acceptance. Hence, we present PrivXR, our two-fold work-in-progress solution: (1) a cross-platform AR/VR privacy panel that provides information on privacy risks, and privacy controls over XR inputs (e.g., VR controllers); and (2) a privacy API in the form of a Unity extension, which is easy for developers to implement, and reads the privacy settings defined by users in the privacy panel. This results in a high-level solution that lets end-users control the XR features of any third-party app which implements the API. The task of making third-party applications compatible with the privacy panel will fall upon XR application developers, who have been known to take the responsibility for users' privacy themselves [14]. Thus, a unified privacy API may also be beneficial for them by streamlining protections. Furthermore, this will alleviate the overhead for end-users, as they will only need to install the privacy panel.

Finally, our approach contributes in laying ground work

979-8-3503-0436-7/24/$31.00 ©2024 IEEE



(a) List of available devices for an XR system, e.g., the headset and controllers on a VR system.

(b) List of XR-related inputs available on the chosen device.

(c) Privacy information and controls and for a given input.

Fig. 1: Possible design for our privacy panel application.

in usable privacy-preserving solutions for XR environments, which remain scarce at the time of writing. We argue that expanding usable privacy research to the field of XR will be beneficial in the long-term, notably by fostering user-centered design aspects in order to inform and support privacy [15]. User-centered design is also typically beneficial for improving the privacy and security of end-users [16]. Therefore, through user-centered design, we hope for our work to become a viable alternative to enhance privacy in XR systems.

In the following, we introduce our privacy panel in Sec. II. We then follow with a description of the associated privacy API in Sec. III. Finally, we conclude with a discussion of the challenges and future work for this project in Sec. IV.

II. Privacy Panel

We first present the end-user side of our solution, in the form of a privacy panel that runs in a dedicated application. We showcase the application with a new user-friendly design shown in Fig. 1, which will eventually replace the current prototype. In the current implementation, users are first shown a list of XR devices present in Unity's XR environment, e.g., headsets and controllers (Fig. 1a). When clicking on a device in the list, the different inputs (or XR features) are shown (Fig. 1b). On systems that contain only one device, e.g., an AR-capable smartphone, the list of devices is hidden, and users are directly shown the XR features.

When clicking on a given input, a third window provides information about the related privacy risks, and privacy controls over compatible third-party applications (Fig. 1c). The informational section lists possible attacks on the selected input. The privacy settings section lets users enable or disable the access to the input, through a toggle button. Disabling inputs that are not required by certain applications can help users ensure that such inputs will not be used in an unlawful way. However, a simple ”on/off” setting remains a compromising choice in most cases for users. Most XR applications require the position and rotation of the device they run on, and of the user's hands or controllers. Disabling these inputs will often result in reduced functionality at best, and render applications completely unusable at worst. Therefore, we also provide an ”input disguise” function, which introduces noise to input data. The introduction of such noise to the data is expected to reduce the efficiency of machine learning models, which are already able to, e.g., (re)identify users based on body tracking data [5], [17]. In order to explain this concept in a user-friendly way, more information is given when clicking the information icon next to the feature, as seen in Fig. 1c.

We provide users with three levels of noise intensity, ranging from low to high. This lets them lower or increase the amount of noise applied to the input depending on their use case. For example, a left-handed user interacting in a collaborative VR drawing application may want to use low intensity disguise on the left controller, in order to draw precisely, and use high intensity on the right controller, as it is not used to draw.

III. Privacy API

A. Integration within the Unity Ecosystem

1) Design Decisions: We designed our privacy API in the form of a Unity extension for three main reasons. Firstly, it is currently impossible to access low-level components of most XR devices, due to device vendors locking up their operating systems for security reasons. Furthermore, it is challenging to obtain sensor usage information from another developer's third-party application. These constraints leave few possible approaches, one of them being application patching, as done in [9]. Thus, we decided to interface our API between the third-party applications that implement the API, and the XR *Software Development Kit*s (SDKs), in order to let end-users control the access to sensor data. Secondly, Unity is one of the most widely used game engines on the market [18], which implies that privacy solutions developed for Unity may more easily cover a wide range of end-users. Unity is also a popular choice for XR development, and thus fits our use case perfectly. Thirdly, this approach is relatively easy for third party developers to adopt: they only need to import the Unity extension, and replace function calls to Unity's *XR Interaction* (XRI) with function calls to our API. It then returns data depending on user choices, which are defined in a separate application that they do not need to bundle. This way, end-users only have to install the privacy panel, which could realistically be distributed on app stores, and developers only need to embed an extension within their applications to make them comply with our solution. Nevertheless, we acknowledge that this approach is only viable if developers accept this solution and use it in their XR experiences.

2) Resulting Design: The architecture of our privacy panel and privacy API in its current implementation is shown in Fig. 2. We gain control over the XR features by overriding scripts of Unity's XRI package, which are called by the main code of the application (i.e., the third party developer's code). We can then enable, disable, and modify data related to these features. To do this, the API communicates with the privacy panel to fetch user settings. Choices made by users in terms of sensor access and sensor noise are applied during the third-party application's runtime.

Cross-Platform Nature. Unity supports building applications for virtually any platform. Also, its device-agnostic XR SDK—the XRI package—allows developers to create cross-platform applications, where multiple users can interact in real-time while using different XR technologies, e.g., mobile AR, VR headsets, etc. Because our API is a Unity extension that interfaces with XRI, it is therefore also compatible with different platforms and devices.

XR Features Access Control. We enable users to disable XR-related features that they may not want applications to access, such as controller position. When access to a given feature is disabled in the privacy panel, its database is updated. When a compliant application uses the API to access this feature, it queries the privacy panel database to check whether access is allowed. Currently, we implemented a simple caching mechanism (without invalidation), so that the API only receives the current setting once when the application first request access to a feature.

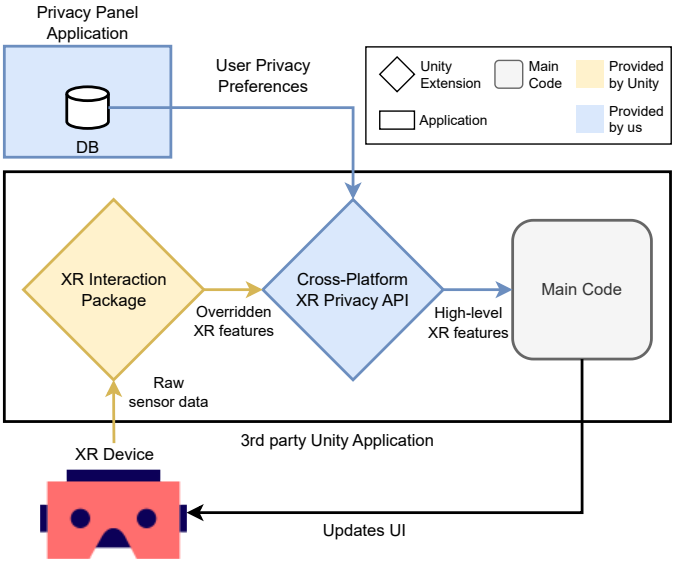


Fig. 2: Architecture of a Unity application implementing our XR Privacy API.

Input Disguise. Our API supports the addition of noise to sensor data, e.g., the position and rotation of VR controllers. This enables users to keep functionality, while enhancing privacy. However, adding too much noise will disrupt the quality of experience of users, e.g., when aiming with precision or performing a specific gesture. Therefore, a balance between privacy and usability is required. For this, we implemented our noise function in a dynamic way: the amount of noise is scaled to the velocity of the input. For example, noise added to the position data of a VR controller will increase in intensity when the movements are fast and ample, but will decrease when the movements are slow and precise. This way, users can still perform tasks that require precision.

IV. Conclusion and Future Work

We have presented our work-in-progress solution that aims to provide privacy transparency and control to end-users in XR environments. Future work will focus on adding more functionalities, conducting user studies to evaluate the user interface, and attempting to address a number of remaining challenges.

Providing sensor usage information to users. Although XR feature control is functional, a user-friendly way of showing sensor usage with the privacy panel remains to be found for all supported platforms. Currently, this can only be done by opening the privacy panel application in full screen. Background usage solutions, such as notifications and widgets, are currently being investigated.

Increasing user awareness. We plan to extend the section about privacy risks, with additional explanations and resources, like in orthogonal domains, such as [19]. Also, we may display

input sensor values in a readable way for users. With these changes, we aim to make this privacy panel an informational tool for users, as well as a tool that provides control over privacy. This is however a considerable challenge, as XR devices run on different operating systems, which have varying levels of access to sensor data.

Supporting additional features. We plan to support more sensors and more platforms in the future. Depending on the possibilities, we will explore input disguise for eye trackers and hand trackers. Regarding the noise intensity settings, we are considering supporting custom noise levels per application, for finer tuning of every compatible XR experience. In addition, third party developers implementing our API could provide optimised default values themselves for their applications. Other aspects, such as a better caching mechanism (with invalidation, in order to support privacy settings updates at runtime), are also considered.

Extend support for other XR SDKs and other XR platforms. Currently, our system only supports apps made with Unity's XRI package. We will extend the API to support other SDKs, such as Meta's XR SDK. This would allow support for a wider range of devices, and thus, users. Additionally, we currently only support XR devices running on Android, such as Meta's Quest VR headsets, and AR applications running on Android. This limitation exists because the communication between the API and the privacy panel is currently done by using Android content providers, which enable sharing data between different applications. While this can be acceptable, as Android devices are well represented in the market, we are nonetheless investigating other methods to communicate between the API and the privacy panel. Eventually, we aim to support more XR platforms, e.g. MR headsets running Windows (Hololens), or the upcoming Apple Vision Pro.

Evaluation of usability and user acceptance. Lastly, we are planning user studies to evaluate the interest given by end-users and developers in our solution, and its usability. We have already gone through different iterations in the design of the proposed user interface, through rounds of feedback with our students, and will go through further iterations after submitting it to users in usability studies, thus following a user-centered approach. We are also interested in observing the perceived usefulness and perceived ease of use for our system.

Eventually, upon completion of this work, we aim to provide users with more choices to protect their privacy in XR systems, among the existing solutions. We also hope for our solution and other related privacy-preserving tools to improve over time, by exchanging ideas within the research community. We argue that different approaches to mitigate privacy issues lead to new ideas, and better solutions for end-users. Although functional, our solution can also be seen as a proof-of-concept, in the sense that we show how usable privacy-enhancing technologies could look for XR end-users in the future, should they be implemented by the device vendors directly in the operating system. Thus, this work may also encourage companies to implement more privacy-preserving technologies natively in future generations of XR devices.

References

[1] ARtillery Intelligence, "Extended Reality (XR) Market Size Worldwide from 2021 to 2026 (in Billion U.S. Dollars)." [Online]. Available: www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/

[2] J. A. De Guzman, K. Thilakarathna, and A. Seneviratne, "Security and Privacy Approaches in Mixed Reality: A Literature Survey."

[3] M. Corbett, B. David-John, J. Shang, Y. C. Hu, and B. Ji, "BystandAR: Protecting Bystander Visual Data in Augmented Reality Systems," in *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services (MobiSys)*, pp. 370–382.

[4] J. Liebers, S. Brockel, U. Gruenefeld, and S. Schneegass, "Identifying Users by Their Hand Tracking Data in Augmented and Virtual Reality."

[5] V. Nair, W. Guo, J. Mattern, R. Wang, J. F. O'Brien, L. Rosenberg, and D. Song, "Unique Identification of 50,000 Virtual Reality Users from Head and Hand Motion Data," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[6] J. Steil, I. Hagestedt, M. X. Huang, and A. Bulling, "Privacy-Aware Eye Tracking using Differential Privacy," in *ACM Symposium on Eye Tracking Research & Applications (ETRA)*.

[7] Y. Kim, S. Goutam, A. Rahmati, and A. Kaufman, "Erebus: Access Control for Augmented Reality Systems," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 929–946.

[8] G. Ramajayam, T. Sun, C. C. Tan, L. Luo, and H. Ling, "Saliency-Aware Privacy Protection in Augmented Reality Systems," in *Proceedings of the First Workshop on Metaverse Systems and Applications*, pp. 1–6.

[9] V. C. Nair, G. Munilla-Garrido, and D. Song, "Going Incognito in the Metaverse: Achieving Theoretically Optimal Privacy-Usability Tradeoffs in VR," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pp. 1–16.

[10] V. Nair, W. Guo, J. F. O'Brien, L. Rosenberg, and D. Song, "Deep Motion Masking for Secure, Usable, and Scalable Real-Time Anonymization of Virtual Reality Motion Data."

[11] C. Slocum, Y. Zhang, N. Abu-Ghazaleh, and J. Chen, "Going through the Motions: AR/VR Keylogging from User Head Motions," in *USENIX Security Symposium*.

[12] J. Auda, U. Gruenefeld, S. Faltaous, S. Mayer, and S. Schneegass, "A Scoping Survey on Cross-reality Systems," pp. 1–38.

[13] C. Warin and D. Reinhardt, "Vision: Usable Privacy for XR in the Era of the Metaverse," in *Proceedings of the 2022 European Symposium on Usable Security*.

[14] D. Adams, A. Bah, C. Barwulor, N. Musaby, K. Pitkin, and E. M. Redmiles, "Ethics Emerging: the Story of Privacy and Security Perceptions in Virtual Reality," in *Symp. on Usable Privacy and Security*.

[15] R. Y. Wong and D. K. Mulligan, "Bringing Design to the Privacy Table: Broadening "Design" in "Privacy by Design" Through the Lens of HCI," in *Proceedings of the 38th SIGCHI Conference on Human Factors in Computing Systems (CHI)*.

[16] C. Reuter, L. L. Iacono, and A. Benlian, "A Quarter Century of Usable Security and Privacy Research: Transparency, Tailorability, and the Road Ahead," *Behaviour & Information Technology*.

[17] J. Liebers, M. Abdelaziz, L. Mecke, A. Saad, J. Auda, U. Gruenefeld, F. Alt, and S. Schneegass, "Understanding User Identification in Virtual Reality through Behavioral Biometrics and the Effect of Body Normalization," in *SIGCHI Conf. on Human Factors in Computing Systems (CHI)*.

[18] TIGA, "TIGA Survey Reveals that Unity 3D Engine Dominates the UK Third Party Engine Market." [Online]. Available: https://www.prnewswire.com/news-releases/tiga-survey-reveals-that-unity-3d-engine-dominates-the-uk-third-party-engine-market-300900187.html

[19] D. Christin, M. Michalak, and M. Hollick, "Raising user awareness about privacy threats in participatory sensing applications through graphical warnings," in *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, 2013, pp. 445–454.