Using Trusted Execution Environments in Two-factor Authentication: comparing approaches

Roland van Rijswijk-Deij[1,2] and Erik Poll[1]

[1]Radboud University Nijmegen, The Netherlands
{rijswijk,erikpoll}@cs.ru.nl
[2]SURFnet bv, Utrecht, The Netherlands

Abstract: Classic two-factor authentication has been around for a long time and has enjoyed success in certain markets (such as the corporate and the banking environment). A reason for this success are the strong security properties, particularly where user interaction is concerned. These properties hinge on a security token being a physically separate device. This paper investigates whether *Trusted Execution Environments* (TEE) can be used to achieve a comparable level of security without the need to have a separate device. To do this, we introduce a model that shows the security properties of user interaction in two-factor authentication. The model is used to examine two TEE technologies, Intel's IPT and ARM TrustZone, revealing that, although it is possible to get close to classic two-factor authentication in terms of user interaction security, both technologies have distinct drawbacks. The model also clearly shows an open problem shared by many TEEs: how to prove to the user that they are dealing with a trusted application when trusted and untrusted applications share the same display.

Keywords: trusted execution environment, Intel Identity Protection Technology, IPT, ARM TrustZone, two-factor authentication

1 Introduction

Two-factor authentication, based on *"something the user knows"* and *"something the user has"*, is a mature technology that has been around for a long time[1]. Classic two-factor authentication technologies[2], based on one-time password or challenge/response algorithms, have favourable properties from a user interaction perspective. Figure 1 shows an abstract model for user interaction in classic two-factor authentication. It shows a security token on the left and the user's regular device (e.g. laptop, tablet, ...) on the right. The model clearly shows the strict physical separation between the trusted environment (token) and the untrusted environment (laptop, etc.). Whenever a user interacts with one of the two devices it is always clear whether they are dealing with a trusted device.

Smart cards (also often used as authentication tokens) are an exception to this model. Most cards lack a display and a means to input data[3]. This means that the user has no (or only

[1]For example, the first RSA SecurID token was introduced in 1987.
[2]For a comprehensive overview of two-factor authentication solutions we refer to [vRvD11].
[3]There are exceptions, e.g. NagraID cards http://www.nidsecurity.com/

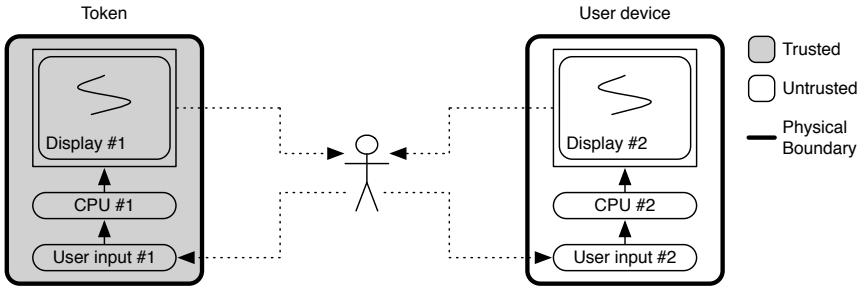

Figure 1: User interaction in classic two-factor authentication

a weak) assurance about the integrity of the data displayed on their screen and about the confidentiality of data entered and sent to the card (e.g. their PIN).

One solution to this problem is to use smart card readers with an integrated keypad for PIN entry and a display. These provide more assurance that the user is interacting directly with the card. A downside is that this requires the reader to be a separate device; this is less attractive because of cost and users needing a reader everywhere they use their card. It also precludes using the readers commonly integrated in modern laptops and smart phones.

In this paper we discuss a different approach to user interaction in two-factor authentication: the use of a *Trusted Execution Environment* (TEE). We investigate if the security model of classic two-factor authentication can be approached for smart cards without the burden of requiring a separate trusted card reader with its own I/O. To do this, we explain what we mean by a Trusted Execution Environment in section 2 and introduce two examples, one from Intel and one from ARM. We then show abstract models for user interaction using these two approaches to a TEE. The paper ends with a comparison of these two approaches and the classic two-factor model and gives directions for future research.

Our contribution We introduce a conceptual model for user interaction with Trusted Execution Environments, which we apply to two concrete TEE technologies (Intel IPT and ARM TrustZone). We show that the model enables us to reason about the security aspects of the interaction between the user and a TEE. The model also clearly illustrates the open problem of how the user can ascertain that they are really dealing with a trusted application on a display that is shared between trusted and untrusted applications.

2 Trusted Execution Environments

Many definitions for a TEE are influenced by the Trusted Computing Group's (TCG) point of view, as the TCG-specified *Trusted Platform Module* (TPM)[4] is the most pervasive approach to trusted computing currently on the market.

[4]http://www.trustedcomputinggroup.org/developers/trusted_platform_module

Vasudevan et al. [VOZ+12] provide a more technology-neutral description, describing a set of features that enable trusted execution[5]. These can be summarised as follows:

- **Isolated Execution** – ensures applications execute completely isolated from and unhindered by others and guarantees that any code and data is protected at run-time.
- **Secure Storage** – protects persistently stored data (e.g. cryptographic keys) belonging to a certain application from being accessed by other applications.
- **Remote Attestation** – enables remote parties to ascertain they are dealing with a particular trusted application on a particular TEE.
- **Secure Provisioning**[6] – enables communication by remote parties with a specific application on a specific TEE while protecting integrity and confidentiality.
- **Trusted Path**[6] – a channel for the user to input data to the TEE and for the TEE to output data to the user; the channel protects against eavesdropping and tampering.

The remainder of this section examines two TEE technologies, Intel's *ITP* and ARM's *TrustZone*.

2.1 Intel Identity Protection Technology (IPT)

It is hard to find technical documentation about IPT. The only public documentation consists of marketing materials and high-level white papers [Int12, Car12, Smi11]. Careful reading of these, however, paints a picture of what IPT is. Intel markets IPT as a number of applications; we describe these below based on Intel's documentation.

One-time Passwords (OTP) The IPT OTP application resembles OTP tokens sold by vendors such as RSA (SecurID) and Vasco (DigiPass). Intel provides a basic implementation based on the OATH time-based OTP algorithm [MMPR11]. Several vendors of classic OTP solutions have also ported their OTP algorithms to IPT (see [Int12], p. 8).

PKI In [Int12] Intel claims that the PKI application[7] introduces hardware protection for RSA keys. The IPT PKI application integrates with Windows applications using a Cryptographic Service Provider (CSP) provided by Intel for Microsoft's CryptoAPI. This is similar to how PKI-enabled smart cards are usually integrated in Windows applications.

[5] Note also that a TEE is much more than just a TPM, which would fulfill only some of the features listed.
[6] Note: secure provisioning is I/O with a *remote* party, a trusted path is *local* secure I/O with the user.
[7] Intel sometimes refers to IPT with PKI as *Platform Embedded Asymmetric Token* (PEAT) (e.g. [Smi11]).

Protected Transaction Display (PTD) PTD is not really an application but rather a feature that supports IPT applications. In documentation Intel describes how this feature can be used to secure PIN entry by the user. The "How It Works" video on Intel's website also shows PTD being used for confirming transactions (e.g. of a bank transfer).

NFC Intel also includes NFC as one of the technologies under the IPT umbrella, but insufficient information is available for us to make any claims about NFC and its relation to IPT, so we have chosen to ignore it in our discussion.

2.1.1 Architecture

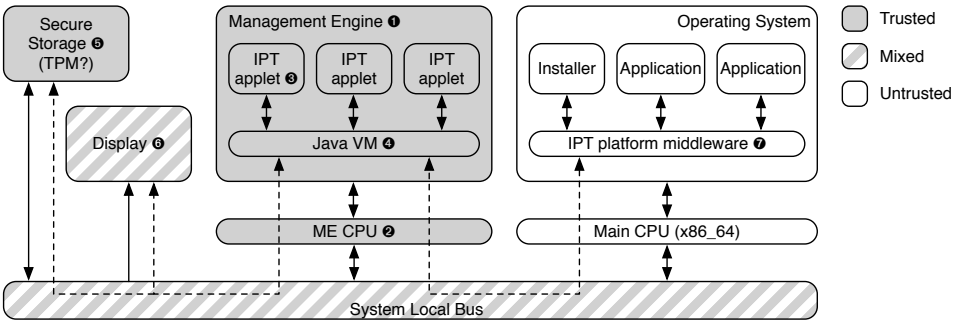

Figure 2: IPT abstract architecture (for a detailed explanation see §2.1.1)

Figure 2 shows an abstract architecture of IPT. It shows the different components identified in Intel's documentation and what environment these components belong to. The paragraphs below provide more detail on each component. Notably absent in this architecture is a trusted path for user input, this is discussed in more detail in section 4.

Management Engine The Management Engine ❶ (ME) appears to be the core of IPT. Based on the naming of the ME it is very likely that Intel re-uses the ME included in their Active Management Technology (AMT)[8]. Assuming this is the case, the ME runs on a separate CPU (an ARC4 RISC processor, shown as ❷ in Figure 2) that runs the Nucleos Real-time OS[9]. IPT applications run as applets ❸ on a Java VM ❹ inside the ME.

Secure Storage ❺ The OTP and PKI application rely on secure storage for key material. It proves difficult to determine if a single subsystem fulfills this function. For OTP Intel [Int12] mentions that one-time passwords are based on a machine-specific key generated by the Intel chipset, but there is no indication of how and where this key is stored. For PKI they [Car12] mention that keys are stored on the hard drive and are wrapped with - what

[8] A technology for remotely managing systems, for instance desktop systems in a large enterprise (http://en.wikipedia.org/wiki/Intel_Active_Management_Technology)

[9] http://www.mentor.com/embedded-software/nucleus/

Intel calls - a Platform Binding Key. All operations on keys then take place in hardware where the key is unwrapped before use. The documentation does not explicitly state this, but it seems likely that the underlying technology used for this is (similar to) a TPM.

Display ❻ It is unclear how the secure display feature integrates with the rest of the system. The examples [Int12, Car12] show that the untrusted OS "sees" black boxes where trusted content is rendered on the screen. This implies that IPT either relies on memory protection for the graphics frame buffer that prevents the untrusted OS from accessing protected parts of the frame buffer, or that the trusted environment has its own frame buffer that is overlaid on frame buffer data from the untrusted OS. It is highly likely that this feature only works with an integrated graphics processor that is part of the chipset.

IPT platform middleware ❼ Communication between applications running in the regular OS on the main CPU and IPT applications in the ME requires some sort of channel. Intel has middleware components that provide such a channel to applications.

Applications that run in the IPT ME can be installed at will. This requires a conduit for installing applications into the ME, a role also performed by the IPT platform middleware.

Attestation and secure provisioning A system with IPT can perform remote attestation to prove that the IPT implementation is genuine using the *Enhanced Privacy Identifier* (EPID) scheme [BL07]. IPT can also set up a mutually authenticated secure channel with the issuer of the attestation identity using Intel's SIGMA protocol [WL11]. This mutually authenticated secure channel can, for instance, be used for secure provisioning.

Developing for IPT As already mentioned, Intel works with independent software vendors to port their OTP solutions to IPT. This implies that there is a software development kit available for IPT. We inquired with Intel as to the availability of an SDK. Intel indicated that such an SDK exists, but that access to the SDK requires a contract with Intel.

IPT and TEE requirements Intel does not market IPT as a TEE. The architecture described above, however, when combined with the description of IPT applications and features in section 2.1, aligns well with the five requirements for TEEs introduced in section 2. Based on this we think that the underlying technology of IPT must be viewed as a TEE.

2.2 ARM TrustZone

ARM offers a technology platform that is similar in its applications to IPT, called TrustZone. Where IPT currently seems to be mostly geared towards use in PC or server class systems, ARM TrustZone is aimed at system-on-a-chip (SoC) architectures used in mobile devices such as smart phones and tablets. This section provides a high-level overview of TrustZone, mostly based on [ARM09].

2.2.1 Architecture

ARM specialises in providing designs for (parts of) so-called Systems-on-a-Chip (SoCs). This is reflected in the TrustZone architecture. The core of TrustZone is a "two worlds" paradigm, with a *normal world* and a *secure world*. This concept shows up all through the architecture. At the hardware level the two worlds are separated on the system bus. What is in effect a special 33rd address line on the bus determines whether bus transactions are part of either one of the worlds. Devices connected to the bus set this address line during a read or write action to indicate whether they are operating in the normal or the secure world. The bus mediates access from bus masters to slaves such that a secure master may access both secure as well as normal slaves whereas a normal master may only access normal slaves and will trigger a bus error if it attempts to access a secure slave.

ARM has also created extensions to its CPU cores called ARM Security Extensions. These allow a single CPU core to run both normal world software and secure world software. Figure 3 shows an abstract model of the Security Extensions. Switching between the two security worlds is managed by the *monitor*, a process that runs in the secure world. The monitor process can be entered by a number of triggers, either programmatically (by executing a special instruction) or by a number of hardware triggers such as interrupts.

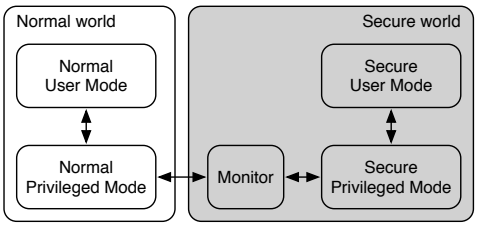

Figure 3: Security Extensions abstract model

2.2.2 Software and TrustZone

ARM does not directly provide any software to execute in the secure world. Developers of systems based on ARM IP either have to develop their own solutions or can choose to use existing secure micro kernels like MobiCore from Trustonic[10]. Trustonic has recently certified that its secure μ-kernel implementation meets the Global Platform Trusted Execution Environment specifications[11,12].

There are also efforts to create open source secure μ-kernels that use the capabilities of TrustZone. Especially worthwhile are the efforts of IAIK (part of the TU Graz). In [Win08] they propose a framework for secure applications on top of TrustZone by executing a modified Linux kernel in the secure world. They also propose an open source development environment for TrustZone [WWPT12] and their own μ-kernel on top of a cheap development board with a Samsung SoC [Win12].

[10] http://www.trustonic.com/about-us/who-we-are/
[11] http://globalplatform.org/specificationsdevice.asp
[12] http://www.trustonic.com/news/release/trustonic-is-first-to-qualify-a-globalplatform-compliant-tee/en

2.2.3 TrustZone and TEE requirements

The list below revisits the requirements for a TEE from section 2 and examines how TrustZone meets these requirements and where additional effort by SoC designers is required:

- **Isolated Execution** – the ARM Security Extensions allow separation of a CPU core into a secure and a none secure world. That in itself is insufficient to provide isolated execution; a secure μ-kernel that supports isolated execution and a memory management unit in the SoC that supports memory protection are also required.
- **Secure Storage** – TrustZone does not include any means for secure storage. Adding something like a Secure Element or a TPM to the SoC design can address this.
- **Remote Attestation** – TrustZone does not provide remote attestation capabilities. This requirement can be fulfilled by introducing a *Mobile Trusted Module* (MTM) [EK07], implemented in hardware (SE/TPM) or in software (in the secure μ-kernel).
- **Secure Provisioning** – Again, this is not explicitly specified as a part of TrustZone, but would most likely be implemented in the secure world μ-kernel.
- **Trusted Path** – Establishing a trusted path is addressed explicitly in TrustZone. In section 3.2 of [ARM09] ARM explains how the bridge between the peripheral bus and the system bus can be used to secure interaction with peripherals like a keyboard. In the example system design in the same document ARM also makes suggestions how the same can be achieved for the display.

3 Related work

Much of the research into trusted execution focuses on aspects of TPMs and cryptographic means to support trusted execution (e.g. attestation). Specific references are not provided as it is easy to find entries into the large body of work around this topic.

Section 2 already references the work by Vasudevan et al. In addition to providing a good definition for a TEE, they argue that TEE facilities are mostly not available to application developers for various reasons, and give recommendations on how to improve this situation. Zhou et al. [ZGNM12] outline an approach for establishing a trusted I/O path between the user and an application on commodity `x86` hardware by proposing modifications to the system's I/O architecture.

Finally, there are two implementations of authentication tokens that mimic the behaviour of a PKI-enabled smart card inside a TEE. Brasser et al. [BBFS12] demonstrate a token running on the user's PC on top of Intel TXT. Tamrakar et al. [TEL+11] take a different approach and emulate a smart card on a smart phone that can interact with a PC as if it were a real smart card.

4 Models for secure user interaction using TEEs

In section 1 we introduced an abstract model for user interaction in classic two-factor authentication (Figure 1), which shows the clear, physical, separation between the trusted and the untrusted environment. In this section we construct similar models based on Intel IPT and ARM TrustZone as TEEs. The models clearly illustrate how IPT and TrustZone differ from the classic approach and also highlight the common issue shared by any approach using a TEE: how to convince the user that they are interacting with a TEE. Note that we do not address securing communication between a TEE and a smart card; existing secure channel solutions provide sufficient means to achieve this.

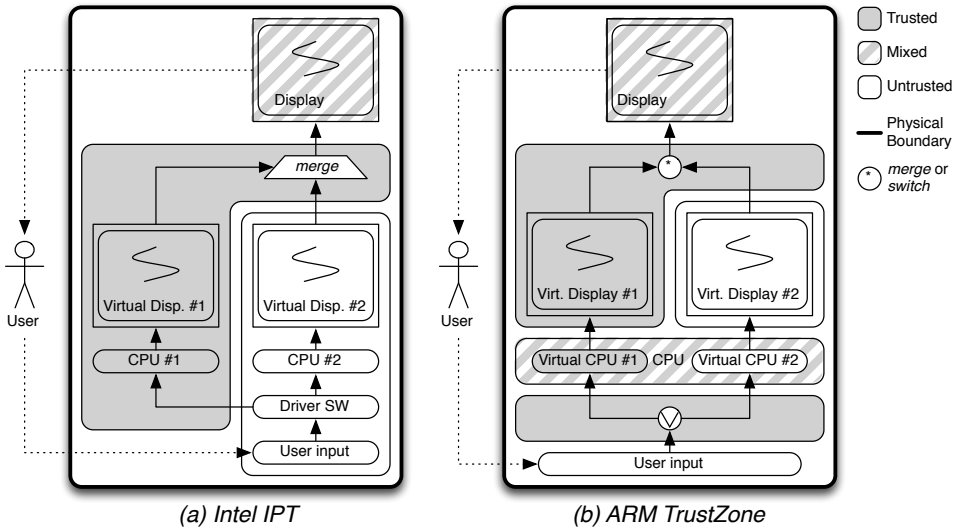

Figure 4: Models for user interaction

4.1 Intel IPT

Based on the features Intel markets under the IPT umbrella (see section 2.1) we have constructed the model shown in Figure 4a. The model shows the trusted environment in gray, the untrusted environment (i.e. the normal OS) in white and components that are in a sense part of both worlds in interleaved gray and white.

The model clearly shows the weakest link in the chain when using IPT: user input does not flow through a trusted path. This is best illustrated by how Intel implements its Protected Transaction Display feature. For PIN entry, the software running in the trusted environment randomises the layout of the PIN entry pad. This is done to prevent applications running in the regular operating system from recording mouse clicks to steal the PIN.

The display at the top of the model is shaded to indicate that it contains content from both

the trusted as well as the untrusted environment. We assume that merging of secure and non-secure elements on the display takes place under supervision of the secure environment (although this is not explicitly stated in the available Intel documentation).

4.2 ARM TrustZone

Figure 4b shows a similar model for ARM TrustZone. Because ARM TrustZone is a set of building blocks and not a stand-alone technology, we have made assumptions (reflecting the most desirable situation that can be created using TrustZone) about the specific configuration, namely

- there is a trusted path to the display, e.g. as suggested in section 3.2 of [ARM09];
- all user input goes through a TrustZone-aware peripheral bus;
- there is a *Memory Management Unit* (MMU) that supports protected memory separation between the secure and normal world.

Under these assumptions the model shows that a fully trusted path can be created all the way from user input to output on the display. The model reflects that there may be multiple implementation options for a trusted display; the display may show either content exclusively from the secure world or the normal world (indicated by "switch" in the model), or it may show a mix of the two just like Intel IPT (indicated by "merge" in the model).

4.3 Local attestation

The models highlight that IPT and TrustZone share a common issue: the display is used for communication by both the trusted and the untrusted environment. This makes it hard for users to ascertain whether they are dealing with a trusted application or not. In fact, all trusted execution environments that allow direct user interaction have this problem.

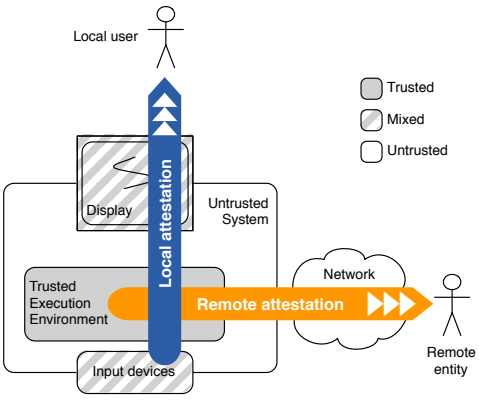

Figure 5: Local versus remote attestation

To remedy this situation the trusted environment will need to provide some form of proof to the user that the data displayed belongs to the TEE and can be trusted. Section 2 mentions remote attestation (proving to remote parties they are dealing with a genuine application and TEE). In keeping with this naming we will call proving trustability to the local user *local attestation*. Figure 5 shows the relation between local and remote attestation.

There are a number of approaches to implementing local attestation. One approach is to set the colour of the title bar of application windows such that all the windows belonging to a single application have the same colour (this approach is taken by Qubes OS[13]). The colour is set by a secure layer of the OS. This approach, however, does not stop malicious applications from spawning windows with content similar to a trusted application. Another approach is *personalisation* of the trusted environment with something specific to the user (e.g. a picture of their family). This personal item is then shown every time the TEE uses the display. The problem with this approach is that it is vulnerable to phishing. The user can, for instance, be tricked into thinking they are reconfiguring their trusted environment and unwittingly submit their personal item to a malicious application. There are also proposals for using a separate trusted device that the user can rely on to perform local attestation of a TEE (e.g. [Toe09, MPSvD07]). Finally, a truly convincing solution is using a hardware indicator on the device that shows the status of the TEE. An example could be an LED that only lights up when the TEE is active. Texas Instruments has submitted a patent application for this [CD02]. Note that this only works well if the entire display is controlled by the TEE.

Neither IPT nor TrustZone provide a clear way to perform local attestation. The examples in Intel's documentation seem to indicate that they hope to achieve this with consistent branding; from a security perspective that has no use, though, since it is trivial for an attacker to observe this branding and to falsify it. TrustZone itself does not address local attestation, but online demonstration videos suggest that Trustonic's MobiCore supports personalisation.

5 Conclusions and future work

Classic two-factor authentication has very desirable security properties but also has practical problems. Users may forget their security token or may lack the infrastructure to use their token (for instance when the token is a smart card that requires a reader). Zooming in on smart cards we already outlined that their security properties are less favourable since they commonly lack a secure display and trusted input device.

We wanted to examine if Trusted Execution Environments can provide secure user interaction similar to classic solutions. It would be particularly interesting if TEEs can also be used to secure interaction with a smart card (given the less favourable properties of a smart card when compared to classic security tokens). To illustrate this we introduced an abstract model for user interaction. We described two TEE technologies (from Intel and ARM) and applied the same abstract model to these two TEE technologies. When we look at how the models for these TEEs compare to the classic model we can conclude that they can approach the classic model up to a certain extent. They do, however, both have significant drawbacks when compared to the classic model. Intel IPT has a serious issue where there is no trusted input path for the user to enter data. ARM TrustZone requires careful selection of the right components by the system-on-a-chip designer that puts the parts of

[13] http://qubes-os.org/

the TEE together to guarantee that it can be trusted. An added disadvantage of TrustZone is that - unlike IPT - it does not come with a dedicated software implementation, further complicating the choices for designers of a TrustZone-based TEE. Finally, both technologies share a common issue, which is how to prove to the user that they are dealing with a trusted application.

It is clear then that these technologies cannot provide a drop-in replacement for classic two factor authentication solutions. This does not mean they do not have their benefits. The convenience of a built-in two-factor authentication solution, such as e.g. Intel IPT can offer, makes it much easier to deploy the solution, thus lowering the threshold for using something that is more secure than the age-old username/password paradigm. Note that a TEE is effectively an embedded smart card, a fact that is capitalised upon by Intel IPT and by the two examples mentioned in section 3. Furthermore TEEs could be leveraged to secure interaction with the user when using smart cards, thus improving the security properties of smart cards when used as a two-factor authentication token. This would also mean that no special secure card reader is required and the built-in smart card readers that appear in more-and-more laptops, tablets and smart phones can be used.

Finally, we note that it proved hard to find detailed public documentation about the specific technologies we investigated, particularly about Intel IPT. Although we feel that this did not impact the conclusions of our research unduly, this is worrisome from a security perspective; public scrutiny is essential for a good understanding and acceptance of these kinds of technologies.

Future work A consortium of partners[14] is currently working on a privacy-friendly authentication technology implemented on smart cards called IRMA[15]. One of the open issues in the project is secure user interaction (both for showing and confirming transaction details and for secure PIN entry). We would like to investigate if a TEE can help solve this issue, which motivated the current paper.

Another question for future research concerns the problem described in Section 4.3: what are alternatives for the personalisation approach that are less likely to be phished?

Finally, it would be worthwhile to investigate and compare the size of the *Trusted Computing Base* (TCB) for IPT and TrustZone-based TEEs, as their security to a large extent depends on the size of the TCB.

References

[ARM09] ARM Ltd. ARM Security Technology - Building a Secure System using TrustZone Technology, 2009.

[BBFS12] F.F. Brasser, S. Bugiel, A. Filyanov, and A. Sadeghi. Softer Smartcards - Usable Cryptographic Tokens with Secure Execution. In *Financial Cryptography and Data Security*, vol. 7397 of *LNCS*, pp 329–343. Springer, 2012.

[14] TNO (http://www.tno.nl), SURFnet (http://www.surfnet.nl) and SIDN (http://www.sidn.nl)

[15] https://www.irmacard.org/

[BL07] E. Brickell and J. Li. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. *IEEE Transactions On Dependable And Secure Computing*, 9(3):21–30, 2007.

[Car12] P. Carbin. Intel Identity Protection Technology with PKI (Intel IPT with PKI) Technology Overview, 2012.

[CD02] B. Cornillault and F. Dahan. Secure Mode Indicator for Smart Phone or PDA, 2002.

[EK07] JE Ekberg and M. Kylänpäa. Mobile Trusted Module (MTM) - an introduction. Technical report, Nokia, 2007.

[Int12] Intel. Deeper Levels of Security with Intel Identity Protection Technology, 2012.

[MMPR11] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. RFC 6238 - TOTP: Time-based One-Time Password Algorithm, 2011.

[MPSvD07] J.M. McCune, A. Perrig, A. Seshadri, and L. van Doorn. Turtles all the way down: Research challenges in user-based attestation. In *Proceedings of HotSec*. USENIX Association, 2007.

[Smi11] N. Smith. Identity Protection Technology (presentation). In *2011 Kerberos Conference*, Cambridge, MA, 2011. Intel.

[TEL+11] S. Tamrakar, JE Ekberg, P. Laitinen, N Asokan, and T Aura. Can hand-held computers still be better smart cards? In *INTRUST 2010*, vol. 6802 of *LNCS*, pp 200–218. Springer, 2011.

[Toe09] R. Toegl. Tagging the turtle: local attestation for kiosk computing. In *Advances in Information Security and Assurance*, vol. 5576 of *LNCS*, pp 60–69. Springer, 2009.

[VOZ+12] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J.M. McCune. Trustworthy Execution on Mobile Devices: What security properties can my mobile platform give me? In *Trust and Trustworthy Computing*, vol. 7344 of *LNCS*, pp 159–178. Springer, 2012.

[vRvD11] R.M. van Rijswijk and J. van Dijk. tiqr : a novel take on two-factor authentication. In *Proceedings of LISA '11: 25th Large Installation System Administration Conference*, pp 81–97, Boston, MA, 2011. USENIX Association.

[Win08] J. Winter. Trusted computing building blocks for embedded linux-based ARM trustzone platforms. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing - STC '08*, pp 21–30. ACM Press, 2008.

[Win12] J. Winter. Experimenting with ARM TrustZone – Or: How I Met Friendly Piece of Trusted Hardware. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp 1161–1166. IEEE, 2012.

[WL11] J. Walker and J. Li. Key Exchange with Anonymous Authentication Using DAA-SIGMA Protocol. In *INTRUST 2010*, vol. 6802 of *LNCS*, pp 108–127. Springer, 2011.

[WWPT12] J. Winter, P. Wiegele, M. Pirker, and R. Toegl. A Flexible Software Development and Emulation Framework for ARM TrustZone. In *Trusted Systems*, vol. 7222 of *LNCS*, pp 1–15. Springer, 2012.

[ZGNM12] Z. Zhou, V.D. Gligor, J. Newsome, and J.M. McCune. Building Verifiable Trusted Path on Commodity x86 Computers. In *2012 IEEE Symposium on Security and Privacy*, pp 616–630. IEEE, 2012.