Trusted Computing to Increase Security and Privacy in eID Authentication

Jan Vossaert, Jorn Lapon, Bart De Decker, Vincent Naessens

Trusted Computing to Increase Security and Privacy in eID Authentication

Jan Vossaert[1], Jorn Lapon[1], Bart De Decker[2], and Vincent Naessens[1]

[1] KU Leuven, Department of Computer Science, Technology Campus Ghent
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
firstname.lastname at cs.kuleuven.be
[2] KU Leuven, Department of Computer Science, iMinds-DistriNet
Celestijnenlaan 200A, 3001 Heverlee, Belgium
firstname.lastname at cs.kuleuven.be

Abstract. Smart cards are popular devices for storing authentication credentials, because they are easily (trans)portable and offer a secure way for storing these credentials. They have, however, a few disadvantages. First, most smart cards do not have a user interface. Hence, if the smart card requires a PIN, users typically have to enter it via an untrusted workstation. Second, smart cards are resource constrained devices which impedes the adoption of advanced privacy-enhancing technologies (PETs) such as anonymous credentials.
This paper presents a new solution that addresses these issues. It allows users to enter their PIN via the workstation and securely transfer it to the smart card. The solution further extends existing smart card assisted authentication technology based on X.509 credentials with privacy-preserving features such as multi-show unlinkability and selective disclosure. The system can, hence, be used to improve the privacy properties of these rolled-out infrastructures. The solution relies on a secure execution environment running on the workstation. We have put our solution into practice and implemented a prototype.

1 Introduction

With an increasing number of online services, the need for reliable secure authentication grows ever stronger. Hence, many governments are issuing eID cards that enable citizens to authenticate and prove several personal properties. This allows the user to establish a secure authenticated session with a remote service provider. The remote service provider can control access to his service based on the released information. These eID systems are often implemented using a smart card to protect the credentials of the user.

These systems, however, also have multiple drawbacks. First, as with many smart card based systems, the user typically enters his PIN via the workstation. This allows malware on the workstation to intercept the PIN which may lead to further abuse. Second, many systems use X.509 credential technology to authenticate the user. This type of credential, however, does not offer the same privacy preserving features as anonymous credential systems.

This paper presents a strategy that tackles these issues. The *contribution* of this paper is twofold. First, it presents a solution that allows users to securely enter their PIN via a workstation to activate the authentication credentials on their smart card. The solution further extends existing smart card assisted authentication technology based on X.509 credentials with privacy-preserving features such as multi-show unlinkability and selective disclosure. It can, for instance, be used to increase the privacy and security of existing electronic identity infrastructures [10]. It applies the secure virtualization technologies contained on the workstation to realize a Secure Execution Environment (SEE). Second, a prototype of this system is presented to validate our solution. For the prototype, a CCID[3] stack is added to an existing framework for building SEE applications.

2 Related work

Several European countries are issuing governmental eID cards [10]. Many countries use a smart card that contains the credentials of the user. The user enters a PIN to activate the credentials. As most smart card readers do not have a secure pin-pad, the PIN is typically entered via the keyboard of the workstation. This introduces a security risk as the PIN can be intercepted by malware running on the workstation. Many countries use standard X.509 credentials for user authentication. A signed set of attributes is embedded in or linked to the authentication certificate. This approach does not offer the privacy preserving features (i.e. anonymous/pseudonymous authentication and selective disclosure) that anonymous credentials do. A few exceptions, such as the German eID system and the system proposed in [11], do provide similar privacy preserving features.

Anonymous credential systems such as Idemix and U-Prove allow users to authenticate in a privacy preserving manner. Users can choose not to release any information but the fact that they hold a valid credential. Furthermore, they allow users to prove unlinkable provider specific pseudonyms. Users can also select which attributes will be disclosed to the service provider. Several efforts have been done to port these computationally expensive systems to run on a smart card. While in [2, 1] a basic version of the Idemix system is implemented on a smart card, [9] and [13] respectively provide a full implementation of U-Prove and Idemix on a smart card. Although there are still some barriers for using anonymous credential systems (e.g. there is no universally good revocation strategy [6]), these advances make anonymous credential systems an increasingly viable option for rolling out electronic identity infrastructures. However, most existing systems still rely on X.509 credentials. The system presented in this paper can be applied to improve the privacy properties of these existing systems. Systems using anonymous credentials can also benefit from the proposed systems for securely handling the PIN input and for correctly informing the user about the pending transactions (e.g. what information is released to whom).

[3] CCID is a standard for communication between a host and a smart card reader.

3 Background

TCG Trusted Computing. Nowadays, commodity computers are equipped with a Trusted Platform Module (TPM). This is a hardware module attached to the computer's motherboard, extending the system with a set of security features. One of these features is the measurement of the state of the system. To this end, the TPM contains several Platform Configuration Registers (PCRs). These are cleared upon power up and can only be modified using the extend operation, performed inside the TPM. This operation requires two parameters: a PCR register and a value. This value is typically the hash of a binary software component. The result of this operation is a new PCR value, being the hash of the value currently contained in the register and the *value* to extend.

Based on this state, the TPM also supports a number of additional operations. Data can be encrypted with the seal operation and only if the system resides in the state specified during the seal operation can the data be decrypted (unseal). Additionally, the quote command returns a proof of the state (*quote*) which a third party can verify (verifyQuote) asserting that the system runs in a specific (trusted) state. The TPM specification supports both attestation based on an RSA key pair and Direct Anonymous Attestation (DAA) [4]. If the latter is used, no uniquely identifying information is released.

Secure Execution Environment. While TPMs have been built-in in workstations for several years, more recent is the adoption of SEE technologies such as Intel TXT and AMD SVM. These technologies cooperate with the TPM to allow the execution of measured code independently of previously executed software.

McCune et al. presented a framework called *Flicker* [8] that uses these technologies to isolate security critical code from applications and run it in a secure environment. The main, possibly untrusted, OS is temporarily suspended after which the sensitive Piece of Application Logic (*PAL*) is securely executed. When the execution of the sensitive code is completed, the OS resumes execution. The framework supports data transfer between the main OS and the *PAL*. The TPM operations can be used to assert to a remote party that certain data was generated by a trusted *PAL*. In [12] and [3] this framework is respectively extended to allow USB-UHCI and secure user interaction (i.e. input via the keyboard and output via the monitor). The user can only trust the displayed information if he is assured that it has been generated by a trusted *PAL*. Brasser et al. [3] proposed an enrollment procedure during which a user-specific picture is sealed to the state of the trusted *PAL*. This is done in a trusted enrollment environment (e.g. on a freshly installed workstation, not yet connected to the Internet) so that an attacker cannot obtain the picture of the user. Since only the trusted *PAL* can access the picture, the user is assured that the trusted *PAL* is running if the correct picture is shown. In [12] the enrollment procedure is realized using a smartphone on which the user can select his authentication picture. This allows the user to establish trust in a *PAL* running on a workstation that is (possibly) infected by malware or untrusted software.

4 Design

This section lists the different actors, followed by the requirements and a description of the system. Finally, a detailed description of the protocols is presented.

4.1 Roles

We assume a user U working on a workstation that runs a legacy operating system (OS) and supports SEE technologies for running a trusted application (PAL). A smart card reader is attached to the workstation. The user also carries a smart card (C), from a rolled-out eID infrastructure, that contains an X.509 credential. This is used to authenticate the user to a remote service provider (SP). The user needs to unlock the credential on his smart card using his PIN.

4.2 Requirements and adversary model

Requirements

R_1 Malware running on the workstation cannot intercept the PIN of the user.
R_2 Malicious software cannot mislead the user into approving malicious authentication attempts or disclosing more information than desired by the user.
R_3 The user can select which attributes are disclosed to the service provider.
R_4 The service provider is assured that the attributes released during authentication are certified by a trusted CA.

Adversary model We assume an attacker that can manipulate the user's operating system and application. Regarding the secure execution environment, simple hardware attacks (e.g. via DMA, memory dump) are taken into account but invasive, side-channel attacks and shoulder surfing are considered out of scope. With respect to the cryptographic capabilities of the attacker, we follow the Dolev-Yao attacker model: attackers can not break cryptographic primitives, but they can perform protocol-level attacks.

4.3 General approach

The user authenticates towards a remote service provider via a workstation using his smart card. Instead of using the X.509 credentials on the smart card to authenticate towards the service provider, they are used to authenticate towards a trusted PAL running on the workstation. To this end, the public key of the CA (pk_{ca}) is embedded in the PAL binary. The PAL is started to handle the input of the PIN and the authentication. This protects the PIN from being intercepted by malware running on the workstation. The PAL verifies the authentication of the smart card and can, subsequently, selectively attest the obtained attributes towards the service provider using the *quote* functionality of the TPM. Since the TPM's DAA capabilities are used, no information but the disclosed attributes,

the fact that the user had a valid credential and that the filtering was performed by a trusted application is released. This allows multi-show unlinkability if the user doesn't disclose uniquely identifying information. The *PAL* not only acts as an authentication proxy between the eID and *SP*, it also gives the user control over which attributes are released to *SP*. This prevents malware on the workstation from misleading the user into approving malicious transactions.

4.4 Protocols

Prerequisites The SEE technologies on the workstation are enabled and the TPM has been certified (i.e. obtained a DAA credential). The service provider obtained the PCR state of the trusted *PAL*. The state of the *PAL* is certified by a trusted third party. The user has gone through an enrollment phase with the workstation during which his authentication image (img_u) is sealed to the state of the trusted *PAL*, resulting in an encrypted image enc_{img}.

Authentication Figure 1 presents the protocol for authenticating the user towards a remote service provider. First, the user requests access to a protected resource from the service provider (1). The provider responds with an authentication request containing its certificate, the attribute request and an attestation challenge (2). Subsequently, the *PAL* is started and the attribute request is passed as a parameter, together with (enc_{img}) and the certificate of the service provider (3). The *PAL* now unseals enc_{img} (4). The user is subsequently informed by the *PAL* regarding the pending authentication and requested to enter his PIN. Meanwhile, the user's unique image is shown to indicate that the trusted environment is running (5). The user can, hence, trust the information shown on the display. To acknowledge the authentication, the user enters his PIN (6). The *PAL* now unlocks the credentials on the smart card using the PIN and sends a challenge to the smart card (7). If the PIN verification succeeds (8), the card signs the challenge and transfers the resulting signature and the certificate ($cert_u$) to the *PAL* (9-10). The *PAL* now verifies the authentication (11-12). If the verification succeeds, the *PAL* extracts the requested attributes (*atts*) from $cert_u$ (13). The *PAL extends* its state with the requested attributes and the certificate of the service provider (14). The *PAL* ends its execution and returns the user's attributes to the regular OS, that resumes its operation (15). The OS now performs a *quote* operation on the state resulting from the *PAL* execution (16). This quote attests towards *SP* that a trusted *PAL* obtained the user's attributes from a valid smart card (i.e. the *PAL* state is extended with the user's attributes) and that the user was shown the correct information about the service provider (i.e. the *PAL* state is extended with the service provider's certificate). The resulting quote is sent to *SP* along with *atts* (17). The service provider verifies the quote and, hereby, checks the authenticity of the received attributes (18). Upon success, access is granted (19).

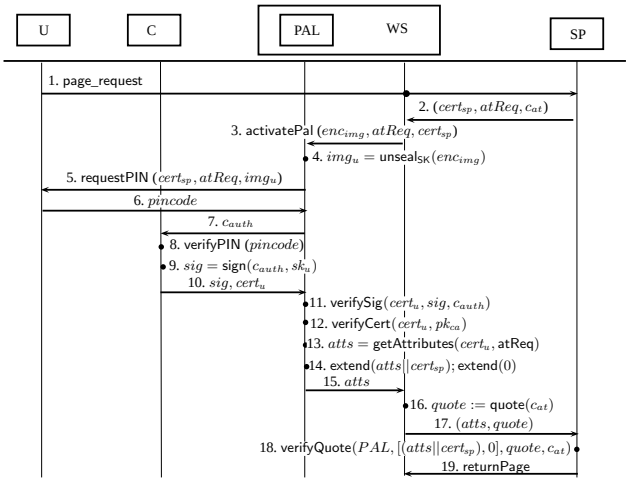

Fig. 1. Privacy friendly authentication using X.509 credentials.

5 Validation

This section validates and illustrates the feasibility of our approach by developing a prototype for the Belgian eID infrastructure [5]. The Belgian eID [5] contains an X.509 credential that is used for user authentication. The middleware implements a `PKCS#11` interface that other applications can use to support eID authentication. Several service providers use the eID infrastructure to handle user authentication. To test the compatibility with these existing service providers, the prototype is forced to release the entire authentication certificate. Adding the privacy-preserving parts of the protocol does not pose any additional technical challenges but requires some additional logic at the service provider.

A smartphone is used in the enrollment procedure during which the user seals an authentication image to the state of the trusted *PAL*. The `PKCS#11` module of the existing middleware has been modified so that the operations requiring a verified PIN are delegated to the trusted *PAL*. For the *PAL* application, a CCID driver was implemented on top of the USB-UHCI stack. This allows the *PAL* to communicate with the eID inserted in the smart card reader. The *PAL* further contains a minimal version of the eID middleware to unlock and use the credentials on the eID.

Our prototype adds a total of 2641 lines of code to the *Flicker* framework, preserving a minimal Trusted Computing Base (TCB). Although adding the selective disclosure part of the protocol will slightly increase the TCB, it will remain sufficiently small to suggest that it can be formally verified. The whole authentication process takes about seven seconds compared to about two seconds for the regular authentication process. The main bottleneck are the TPM operations. Hence, the performance can be increases significantly by using a virtual TPM [7], while maintaining the same security properties.

The prototype has been tested with existing Web-based service providers. No modification of the browser or service provider are required. The privacy-preserving functionality, however, does require adding some additional logic to the service provider. A similar approach can be taken for other European eID cards as many (e.g. Italy, Spain and Portugal) use a design similar to the Belgian eID.

6 Evaluation

Requirements review During authentication the user is assured that the trusted application is running by displaying the personal image. Since the user only enters his PIN if the correct image is shown, the *PAL* is in full control of the workstation during the input of the PIN. Hence, malware running in the OS cannot intercept the PIN (cfr. R_1). Similarly, the user is assured that the provided information about the pending authentication is correct. The *PAL* binds the authentication proof to the service provider shown to the user by extending its state with the provider's certificate (cfr. R_2). During authentication of the eID to the *PAL*, the latter obtains all of the user's attributes contained on the card. The *PAL* filters the received information and only releases the attributes approved by the user (cfr. R_3). The *PAL* verifies the authenticity of the smart card via a challenge-response protocol. The *PAL* only continues the authentication if a valid smart card is used. Via the attestation protocol, the service provider is assured that a trusted *PAL* checked the validity of the smart card and filtered the attributes (cfr. R_4).

Security and privacy considerations The *PAL* is trusted by both the user and the service provider to correctly execute the specified protocol. The functionality of the *PAL* is kept to the minimum. The small TCB decreases the chance of bugs and suggests that formal verification is possible. Moreover, the *PAL* can be updated by distributing a new binary to the workstation and the new certified state to the mobile and providers. Subsequently, the previous version is revoked.

Identity cards that have been stolen or otherwise compromised can be revoked. This has no impact on the *trusted PIN input* system as the service provider receives the authentication certificate and can, hence, check the revocation status. In the *privacy friendly authentication* system, however, the service provider only receives the attributes released by the *PAL*. The *PAL*, therefore, is responsible for checking the revocation status of the card. In case the revocation list contains a limited number of serials, it can be passed along to the *PAL* as an argument. In case the revocation list is too large, the *PAL* generates a nonce that is transferred to an OCSP server together with the serial of the eID. The nonce should be transferred via the regular OS to the OCSP server to ensure that the *PAL* doesn't have to contain networking drivers which would bloat the TCB. The response is subsequently transferred back to the *PAL* that can now verify the revocation status of the card.

7 Conclusion

This paper presented a system that can be applied to increase the security and privacy of smart card based authentication systems that use X.509 credentials. It allows users to securely enter their pincode via the workstation to activate the authentication credentials on the smart card. The system further realizes privacy enhancing features such as multi-show unlinkability and selective disclosure. A prototype using the Belgian eID card is presented, illustrating the feasibility of the system. The system applies SEE technologies included in most workstations.

References

1. Patrik Bichsel, Jan Camenisch, Bart Decker, Jorn Lapon, Vincent Naessens, and Dieter Sommer. Data-minimizing authentication goes mobile. In Communications and Multimedia Security, volume 7394, pages 55–71. Springer, 2012.
2. Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In Proceedings of the 16th ACM conference on Computer and communications security, CCS '09, pages 600–610. ACM, 2009.
3. Franz Ferdinand Brasser, Sven Bugiel, Atanas Filyanov, Ahmad-Reza Sadeghi, and Steffen Schulz. Softer smartcards - usable cryptographic tokens with secure execution. In Financial Cryptography, volume 7397, pages 329–343. Springer, 2012.
4. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS '04, pages 132–145, New York, NY, USA, 2004. ACM.
5. Danny Cock, Karel Wouters, and Bart Preneel. Introduction to the belgian eid card. In Public Key Infrastructure, volume 3093, pages 1–13. Springer, 2004.
6. Jorn Lapon, Markulf Kohlweiss, Bart Decker, and Vincent Naessens. Analysis of revocation strategies for anonymous idemix credentials. In Communications and Multimedia Security, volume 7025, pages 3–17. Springer Berlin Heidelberg, 2011.
7. Jonathan McCune, Yanlin Li, Ning Qu, Zongwei Zhou, Anupam Datta, Virgil Gligor, and Adrian Perrig. Trustvisor: Efficient TCB reduction and attestation. In Proceedings of IEEE Symposium on Security and Privacy, pages 143–158, 2010.
8. Jonathan M. McCune, Bryan J. Parno, Adrian Perrig, Michael K. Reiter, and Hiroshi Isozaki. Flicker: an execution infrastructure for TCB minimization. SIGOPS Oper. Syst. Rev., 42(4):315–328, April 2008.
9. Wojciech Mostowski and Pim Vullers. Efficient U-Prove implementation for anonymous credentials on smart cards. In Security and Privacy in Communication Networks, volume 96, pages 243–260. Springer Berlin Heidelberg, 2012.
10. Ingo Naumann and Giles Hogben. Privacy features of european eID card specifications. `www.enisa.europa.eu`, 2009.
11. Jan Vossaert, Jorn Lapon, Bart Decker, and Vincent Naessens. User-centric identity management using trusted modules. In Public Key Infrastructures, Services and Applications, volume 6711, pages 155–170. Springer Berlin Heidelberg, 2011.
12. Jan Vossaert, Jorn Lapon, Bart Decker, and Vincent Naessens. Client-side biometric verification based on trusted computing. In Communications and Multimedia Security, volume 8099, pages 34–49. Springer Berlin Heidelberg, 2013.
13. Pim Vullers and Gergely Alpár. Efficient selective disclosure on smart cards using Idemix. In Policies and Research in Identity Management, volume 396, pages 53–67. Springer Berlin Heidelberg, 2013.