HAL
archives-ouvertes.fr

The Radboud Reader: A Minimal Trusted Smartcard Reader for Securing Online Transactions

Erik Poll, Joeri Ruiter

▶ **To cite this version:**

Erik Poll, Joeri Ruiter. The Radboud Reader: A Minimal Trusted Smartcard Reader for Securing Online Transactions. Simone Fischer-Hübner; Elisabeth Leeuw; Chris Mitchell. 3rd Policies and Research in Identity Management (IDMAN), Apr 2013, London, United Kingdom. Springer, IFIP Advances in Information and Communication Technology, AICT-396, pp.107-120, 2013, Policies and Research in Identity Management. <10.1007/978-3-642-37282-7_11>. <hal-01470495>

HAL Id: hal-01470495
https://hal.inria.fr/hal-01470495

Submitted on 17 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

The Radboud Reader: a minimal trusted smartcard reader for securing online transactions

Erik Poll and Joeri de Ruiter

Institute for Computing and Information Sciences, Digital Security Group,
Radboud University Nijmegen, The Netherlands.

Abstract. We present the design of a device for securing online transactions, e.g. for internet banking, which can protect against PC malware, including Man-in-the-Browser attacks. The device consists of a USB-connected smartcard reader with a small display and numeric keyboard, similar to devices currently used for internet banking. However, unlike existing devices, we rigorously stick to the design philosophy that the device should be as simple as possible; move functionality and control is moved as much as possible to the smartcard. Although this is a simple (and obvious) idea, we are not aware of any solutions pursuing it. Moreover, it has some interesting benefits compared to existing solutions: the device is simpler, provides stronger security guarantees than many alternatives (namely that it will only display text authenticated by the smartcard), and is generic in that it can be used in combination with different smartcards for different applications (for example, for internet banking with a bank card and for filing an online tax return with a national ID card).

1 Introduction

A fundamental problem in securing online transactions is the lack of a trustworthy device to communicate with the human end user. The software on laptops and PCs, but also smart phones, is so large and complex that security vulnerabilities are inevitable. This means that these devices are not trustworthy as input or output channel to communicate to the user (via the display, keyboard or mouse), as malware can manipulate the display and eavesdrop on any input.

Smartcards are a potential improvement in that they provide a secure computing platform. The chances of exploitable security vulnerabilities in the software on a smartcard are *much* lower than for a PC or laptop, given the very small size and the very limited functionality of the code. However, a serious and fundamental limitation of smartcards is the lack of a display and keyboard.[1] This means that a smartcard requires some terminal with a display for output and a keyboard for user input, and this terminal has to be trusted.

Given the issues above, the more secure solutions that use smartcards to secure online transactions rely on a dedicated smartcard reader with a numeric

[1] Still, the first smartcards with keyboard and display are in commercial use.

keyboard and display. The most prominent example is the EMV-CAP standard for internet banking [3, 10]. Some of these devices are connected by USB to the computer, which can both increase user convenience – as the user does not have to type over numbers to and from the device – and security, by providing a 'What You See is What You Sign' guarantee. Section 5 compares various existing solutions.

This paper presents an alternative design for a solution, the Radboud Reader, where we rigorously stick to the design principle that the device should be as simple as possible. Where possible functionality is moved to the smartcard rather than the reader and the smart card is given control as much as possible. Although these principles are rather obvious, and the resulting system is quite a natural solution, we are not aware of anyone else proposing a system like this. The resulting system which we present in this paper has some interesting advantages over existing solutions:

- The device is *simpler* than alternative solutions: it contains no secrets, and does not need to support any cryptographic operations or even hashing. Keeping the device simple also reduces the chance of security vulnerabilities in the device.
- Our solution gives *stronger security guarantees* than most other solutions: the device provides a trusted display which only displays text approved by the smartcard, so that the smartcard can check say a digital signature on any text before it is displayed.
- The device is *generic* and can be used in combination with different smartcards for different purposes.

Section 2 describes the high-level design of the system, which is discussed in more detail in Section 3. Section 4 discusses our prototype, Section 5 compares our solution with existing devices and Section 6 concludes.

2 Security objectives and high-level design

The basic set-up, shown in Fig. 1, is the same as with any USB-connected smartcard reader for internet banking: a web browser on a PC communicates with a remote web server over the internet and with a smartcard reader via a USB cable. The smartcard reader has its own display and numeric keypad with two additional buttons marked 'OK' and 'Cancel', so the user can interact with both the PC and the reader for input and output, via their respective displays and keyboards.

The objective is that, unlike the PC, the reader can be a trusted input and output device, meaning that we rely on

S1 authenticity of whatever is displayed on the display,
S2 confidentiality of anything that is entered on the keyboard,
S3 non-repudiation of any transactions confirmed or declined by pressing 'OK' or 'Cancel'.

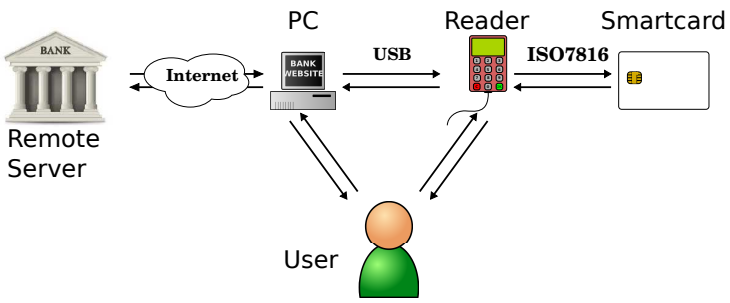

Fig. 1. Set-up

Although a complete transaction might be performed using only the reader, this is not very user friendly given its limited display and numeric-only keyboard. Therefore, the PC can still be used to set up the connection to the service provider and enter the transaction details.

2.1 Attacker model

The attacker is assumed to be in full control of the network and of the PC, including the USB connection to the reader, but not of the reader or the smartcard, and the communication between them. If one abstracts away from the PC then this is equivalent to assuming a Dolev-Yao attacker [2] on the communication between the remote server and the reader.

Our main concern is an online attacker, with possibly total control over the PC, but without physical access to the reader or the smartcard. Attackers with physical access are only a secondary concern. This means that physical tamper-resistance or tamper-evidence of the reader is not crucial: they are nice properties to have, but in practice one will only want to spend a limited amount to realising them to some degree. Of course, one would want to include protection against shoulder surfing, e.g. by not echoing say a PIN code on the display as it is entered on the reader.

2.2 High level design decisions

For the reader to be a trusted input and output device, we will ensure that

- all output on the display comes from the smartcard, and
- all input to the device is only sent to the smartcard,

as illustrated in Fig. 2. This means that the smartcard can ensure authenticity of the output to the screen, and confidentiality of the input from the keyboard.

By only allowing the smartcard to output to the display, we get a trusted display, where authenticity of displayed messages can be enforced by the smartcard. Allowing the PC to display text would allow malware on the PC to control the display, even though the device could still offer the guarantee of 'What You See is What You Sign'.

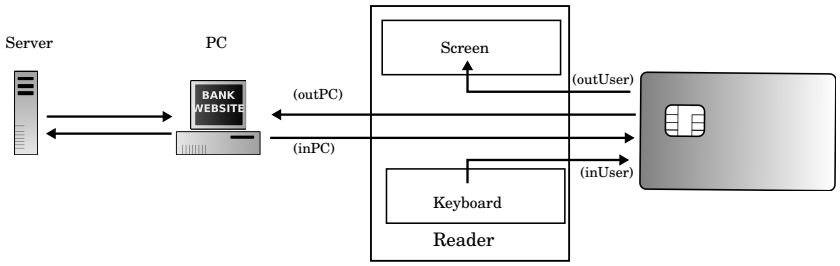

Fig. 2. Information flows in the Radboud Reader. NB all I/O the device offers to the user is with the smartcard, not the PC, and all communication between the user and remote server has to pass through the smartcard.

So *physically* the reader is between the smartcard and the PC, but *logically* – i.e. considering the information flows – the smartcard is between the PC and the reader.[2]

For the input the Radboud Reader provides a numeric keyboard as well as an 'OK' and 'Cancel' button. Output can be displayed on its display consisting of 80 characters (4 x 20).

Note that it is completely up to the smartcard to decide what functionality it provides, and how the data communicated with the PC, and via the PC with the remote server, is secured. Different strategies are possible here:

- The smartcard could set up a secure tunnel to the remote server, analogous to TLS, ensuring integrity and confidentiality of the entire communication session between smartcard and the remote server. In such a set-up, one could let the smartcard *only* communicate with the back-end, and not communicate with the PC at all.
  Many smartcards already provide such a secure tunnelling mechanism, called Secure Messaging in smartcard jargon. The ISO/IEC 7816 standard [9] already describes it, as do many other smartcard standards, including the ICAO standard for electronic passports [8], the EMV standard for payment cards [4], and the Global Platform standard [6].
- Alternatively, one could choose to sign and/or encrypt parts of messages exchanged between the smartcard and the remote server on a more piecemeal basis.

The former approach provides simpler and more robust security. An advantage of the latter approach might be that the browser can see and display some parts of the communication between smartcard and PC, which in the former approach would require additional communication between browser and the back-end.

[2] One could even consider using physically different contacts on the smartcard for communication with the display and communication with the PC. Two of the contacts of smartcard are reserved for future use in the ISO/IEC 7816 standard, so this is possible. However, this would be a more expensive solution requiring non-standard smartcards.

To login to a website using the reader in combination with a smartcard, one could of course let the smartcard generate a credential to login, using some challenge-response protocol. It is even possible to let the smartcard to supply the username (or say, in the case of bank account, the bank account number) to the remote server when logging in over a secure tunnel, in which case malware on the PC could not even be able to learn this. Paper receipts for credit card transaction no longer show the complete card number but only the last four digits. Similarly, when using the Radboud reader in combination with a smartcard to log-on to some website, there is no reason to show the actual login name on the PC's display if leaking this could give useful information to an attacker.

Inserting a smartcard into a smartcard reader that is attached to a (potentially infected) PC is of course dangerous. Malware on the PC could try to access functionality of the smartcard in unwanted ways, for example by sending PIN code guesses to the card, with a small chance of guessing it right, and a big chance of blocking the card with 3 incorrect guesses. This leads to another security requirement, namely that access to functionality of the smartcard is strictly limited, which we will realise by ensuring that

S4 Input from the PC is forwarded to the smartcard in a specific format so that it cannot address arbitrary functionality on the card.

Effectively, the reader should provide a firewall between the smartcard and the PC, which only lets minimal functionality through.

That the device uses a USB-connection to a PC to connect to the internet and the back-end is not essential; any means of connecting the device to the internet could be used. USB is the obvious choice in that it is widely available, and indeed existing smartcard readers use it. An alternative to this would be to connect with the PC via Bluetooth. Another (more expensive) alternative would be to simply equip the device with GSM and let it by-pass the PC completely, though that requires additional measures to link a session on the device with a session on a PC.

2.3 Functional requirements

To achieve the properties discussed previously, the Radboud Reader needs to provide the following operations:

1. Starting a session, by selecting the desired application on the smartcard; a smartcard can hold several applications, and one has to be selected at the start of a session.
2. Forwarding data received from the PC to the smartcard.
3. Carrying out instructions received from the smartcard; these instructions can tell the reader to
   a) display text and wait for the user to press 'OK' or 'Cancel';
   b) display text and wait for user input;
   c) forward data from the smartcard back to the PC, and then wait for new data from the PC.

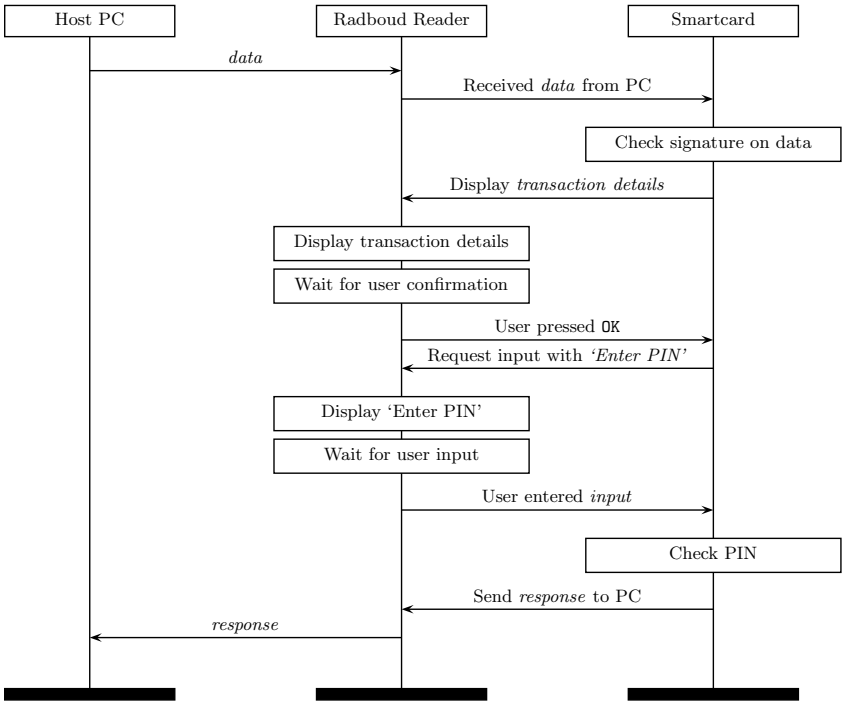

Fig. 3. Typical usage scenario

4. Sending user input from the keyboard to the smartcard, following 3a) or 3b).

Using these operations the smartcard can then construct any interaction with PC and the user that it wants. A typical usage scenario would be that the card receives digitally signed information over the internet and displays this on the device (ensuring that only genuine messages are displayed) and sends out digitally signed responses (to authenticate transactions). Of course, data received and sent can also be encrypted as well as signed to ensure confidentiality. It is up to the smartcard to check or put digital signatures and to en- or decrypt.

Fig. 3 illustrates such a scenario, where the data from the PC is forwarded to the card, who subsequently shows transaction data on the display. If the user agrees with this information, the card requests the PIN code from the user as input. If the PIN code is correct, the card generates a signature which is sent to the reader to be forwarded to the PC.

3 Detailed design

To implement the functional requirements described above, several decisions have to be taken:

(i) How is the applet selected on the smartcard?

(ii) How do we forward data to the smartcard, when data is received from the PC (i.e. inPC in Fig. 2)?
(iii) How is the distinction made between output from the smartcard that is (a) destined for the display, (b) a request for input from the user, and (c) output destined for the PC? (I.e. between outUser and outPC in Fig. 2.)
(iv) How is input by the user forwarded to the smartcard (i.e. inUser in Fig. 2)? And how can the card distinguish between data coming from the keyboard and from the PC? (I.e. between inPC and inUser.)

These operations have to be realised at the level of the communication between the smartcard and the reader, as laid down in the ISO/IEC 7816 standard. At this level, all communication is via data packets called APDUs (Application Protocol Data Units). An APDU is simply a sequence of bytes in a fixed format. The smartcard is a slave in the master-slave communication between smartcard and reader: the reader sends a command to the smartcard, a so-called *command APDU*, and the smartcard answers with a *response APDU*.

The reader will have to present data to the smartcard in such a way that the smartcard can distinguish between inPC and inUser in Fig. 2. Similarly, the smartcard will have to provide its response in such a way that the reader can distinguish between options 3a), 3b) and 3c) listed earlier. Because we want the smartcard to be in control as much as possible, these responses from the smartcard will in fact be instructions [3] for the Radboud reader.

Selecting an application on the smartcard The ISO/IEC 7816 standard uses AIDs (Application IDentifiers) to identify applications on a smartcard. An AID is simply a sequence of bytes of a fixed length. AIDs for certain applications have been standardised, and a smartcard can have one application that is selected by default.

Selecting the desired application could be done in several ways: by fixing a unique AID that is selected and hard-coding this in the reader, by hard-coding a list of AIDs that the reader attempts to select, or by letting the reader select the default applet. Letting the PC choose the applet to select is of course not a good option, as it could be abused by malware on the PC.

For simplicity, we choose to use a fixed AID to select the application on the smartcard. We choose an AID here that is not already used for an existing standard application. As soon as the smartcard is inserted in the reader, the application is selected.

Forwarding PC communications to the smartcard One security-critical piece of functionality of the device is to provide a kind of firewall to shield the smartcard from malicious actions by the PC. One way of doing this would be to block all traffic except a minimal white-list or a single instruction. Another would be to prepend data sent to the smartcard with a fixed prefix.

[3] To avoid confusion with the standard terminology of commands for messages from the reader to the smartcard, we will call such messages from the smartcard to the reader *instructions* and not commands.

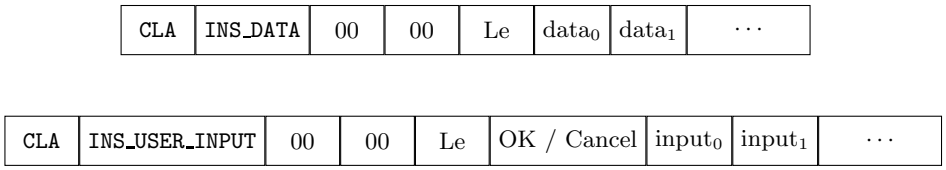

Fig. 4. Data formats for the command APDUs to pass PC data and user input to the card.

The former approach is a classical method for firewalls. The PC can send APDUs to the reader, who will forward them to the card only they are allowed according to its rules. These rules would have to be fixed for all possible applications, as they are stored in the reader. (One could make this APDU firewall configurable, with configuration controlled by the smartcard, but this introduces a lot of complexity.) Therefore it needs to be decided in advance what APDUs might be necessary and can be allowed to pass through.

We choose the latter approach, where the data received from the PC is wrapped as payload in an APDU with a dedicated instruction. Using this approach, the reader does not have to process the data it receives in any way: it simply forwards communication received from the terminal with a fixed prefix.

We are free to choose the prefix, but we should make sure this prefix does not have a meaning already in the ISO/IEC 7816 standard. Otherwise the reader could accidentally trigger functionality in a smartcard that was not designed to be used with the USB reader.

3.1 Format of data sent to the smartcard

As explained above, the reader will simply forward data from the PC to the smartcard – i.e. inPC in Fig. 2 – with a fixed prefix. For user input on the device – inUser– the reader will use a different prefix, so that the card can distinguish them. Fig. 4 gives the data formats for this, which use the constants `INS_DATA` and `INS_USER_INPUT` as the so-called instruction byte (the second byte of the command APDUs):

- `INS_DATA` indicates that the APDU contains data forwarded from the PC. The data that is received from the PC is wrapped in an APDU and forwarded as it is.
- `INS_USER_INPUT` indicates that the APDU contains user input entered on the keyboard. The first byte of the data indicates whether the 'OK' or 'Cancel' button was pressed and, in case that 'OK' was pressed, the rest of the data gives the user input.

3.2 Format of data sent by the smartcard

The three instructions that the smartcard can give to the reader use the data formats presented in Fig. 5, where the first byte specifies the instruction:

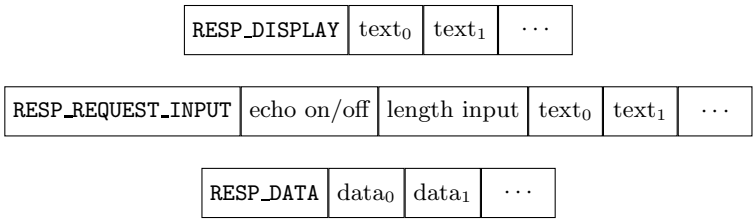

Fig. 5. Data formats for the response APDUs that provide instructions of the smartcard to the reader

- `RESP_DISPLAY` instructs the reader to display the text returned by the smartcard. The text to be displayed should not be longer than 80 characters for our prototype and is supplied after the `RESP_DISPLAY` instruction. When displaying data following a `RESP_DISPLAY` instruction, the reader waits for the user to either press 'OK' or 'Cancel'. This input is then sent to the smartcard using the `INS_USER_INPUT` command.
- `RESP_REQUEST_INPUT` instructs the reader to request user input. The second byte indicates whether input should be echoed, i.e. if the user can see the input on the screen or only masked characters. The third byte indicates the maximum input length. A string is appended to this that will be shown before the input. The length of this string together with the maximum input length cannot be longer than 80 characters for our prototype. After receiving a `RESP_REQUEST_INPUT` instruction, the reader lets the user input data using the keypad. The reader uses a `INS_USER_INPUT` instruction to return the result to the smartcard.
- `RESP_DATA` instructs the reader to forward the data returned by the smartcard to the PC and wait for new data from the PC.

After each command sent to the smartcard, the card can respond with a new instruction, making it possible to perform multiple `RESP_DISPLAY` and `RESP_REQUEST_INPUT` instructions before finally returning data to the PC using the `RESP_DATA` instruction.

4 Prototype reader

To show the functionality of the Radboud Reader, we constructed a prototype using the Arduino platform[4] (see Fig. 6).[5] The code is approximately 350 lines of code and the components cost around 50 Euro. This prototype can demonstrate the feasibility of the approach and developing it was a useful exercise to make sure we resolved all implementation and design choices that have to be made in realising an implementation. Using this prototype protocols can be developed and tested.

[4] http://www.arduino.cc
[5] Available from http://www.cs.ru.nl/ joeri/

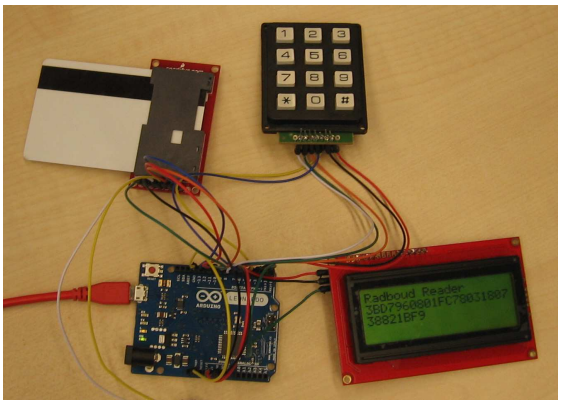

Fig. 6. Prototype based on Arduino platform

We implemented one sample protocol on an actual smartcard. The protocol provides integrity and confidentiality of communication between the smartcard and the issuer. For this, the smartcard contains a symmetric key, that is shared with the issuer, an asymmetric key pair and the public key of the issuer. The issuer also knows the shared symmetric key, the public key of the smartcard and its own asymmetric key pair.

The issuer starts the protocol by sending a command to initialise the protocol. The smartcard responds by sending a nonce and his identity, encrypted using the public key of the issuer. In response the issuer sends the smartcard's nonce together with his own nonce, both encrypted using the public key of the smartcard, back. The smartcard will then generate a random session key, which he encrypts using his shared symmetric key. To ensure integrity of the session key, as it is a random string, a MAC is computed over the encryption.

After the session key is established, so-called Secure Messaging is used to provide confidentiality and integrity of the communication session between the smartcard and issuer, as in an SSH or TLS tunnel.

The Session Sequence Counter (SSC), that is used to prevent replay, is initialised to the concatenation of the nonces of the issuer and the smartcard.

With Secure Messaging, first the data is encrypted using the session key. Subsequently a MAC is computed over the encrypted data. To prevent replays, the first block that is processed in the computation of the MAC is the SSC, whose value is increased by one after each message.

5 Related work – Differences with existing devices

To compare our approach with existing devices, several security features can be taken into consideration:

1. The device may guarantee 'What You See is What You Sign' (i.e. guarantee that the transaction details shown on the display are in fact what is signed),

but it may also provide a trusted display that can guarantee that any data shown on the display is authentic (i.e. originates from the service provider).
2. The device can support cryptographic operations, increasing the complexity of the device.
3. The device can include secrets, for instance in the form of secret keys (in case that the device can do cryptographic operations). But it is also possible that the functionality of the device is (partly) secret, in which case we may not be able to tell if it contains say a secret key or if it simply contains some unknown use of a hash function.
4. The device may be unconnected, and not provide any means of communication between it and the PC, or, in case it does, this communication may be bidirectional (e.g., in the case of USB), or uni-directional (e.g., in the case of optic communication from the PC to the device).
5. Finally, there is the question of how generic the device is, i.e. whether the device can be used for several purposes and users.

For some of these characteristics one can still argue whether they improve or weaken security. E.g., having secrets in the device makes it harder to produce a fake version or a software implementation of the devices, but makes generic use harder.

Table 1 gives a comparison of different devices based on these features. Below we discuss these devices in more detail.

Manufacturers typically do not publish technical details about the devices they produce, (though they do sometimes apply for patents, e.g. [7]). This means that for some devices we do not know all the features, unless the working has been reverse-engineered. Ideally vendors would provide this kind of information publicly so the clients can make a better comparison between different devices.

Unconnected smartcard readers with a display and keyboard are widely used for online banking. Many of these systems use EMV-CAP, a proprietary protocol of MasterCard, defined on top of the EMV standard [4]. This has advantages, namely that existing EMV smartcard implementations, which may have undergone costly security certifications, can be re-used, but also introduces the risk of ambiguities in different meanings of the same protocol messages [3]. When using these devices, the user receives a numeric challenge from the bank on his PC. This challenge and PIN code have to be entered on the reader, which then generates a response, using the user's smartcard, that has to be entered on the PC again. The challenges are often just random numbers with no meaning to the human user. They may also include account numbers or amounts, i.e. data more meaningful to the user, which then could protect against Man-in-the-Browser attacks at the expense of the user having to enter more input and relying on the user to know what the meaning of all parts of the challenge is.

EMV-CAP has been largely reverse-engineered [3, 10]. In some variants the digital signature (or rather, a 3DES MAC) is computed in the device, and not by the smartcard [10]. Note that not having any cryptographic capabilities in the device, as we do, makes such a bad choice impossible.

Some banks already use a USB-connected reader for internet banking. Companies supplying solutions for this include VASCO and Gemalto. As far as we know, none of these devices provide a trusted display: they display data that is received over the USB cable in plaintext without any integrity checks. This means that malware on an infected PC could show anything it wants, and could even use the display of the reader as part of a sophisticated phishing attack. Even if these devices do not guarantee that what is shown on the display is authentic in any way, they can provide 'What You See is What You Sign' by sending the displayed text to the card to be signed.

One variant of a USB-connected reader, produced by Gemalto, has been reverse-engineered, and found to contain a security flaw [1]. This demonstrates once again the danger in relying on closed, proprietary solutions. It also provides further support for our design philosophy of keeping the device as simple as possible.

The Zone Trusted Information Channel, or ZTIC[6] from IBM comes closer to our approach in the security it provides [11]. The ZTIC is a small USB-connect smartcard reader with a small display and allows user input by means of two buttons (for OK and Not OK) and a wheel that can be turned to input numbers. Unlike the Radboud Reader, the ZTIC has cryptographic capabilities and the keys and certificates to set up a secure SSL/TLS tunnel between the ZTIC and a remote web server. Every device has its own certificates for mutual authentication with the issuer.

One difference between the ZTIC and our proposal is that the ZTIC is a more complicated device, capable of storing keys and doing crypto. Unlike our solution, which is generic and can be used in conjunction with any compatible smartcard, the ZTIC needs to have the certificate for each service provider in order to communicate with it. Another difference, and possible advantage of the ZTIC, is that the common functionality to provide a secure tunnel is provided by the ZTIC, and need not be provided by each smartcard used with the Radboud Reader: using the ZTIC it is guaranteed to have a secure tunnel, using the Radboud Reader this still depends on the smartcard.

The FINREAD project proposed a standardised trusted card reader [5]. This idea never became a success, probably because the FINREAD card reader would be too expensive and complex; they were meant to be tamper-resistant and included a PKI support for controlling applications on them.

The AGSES card reader[7] does not use a USB connection, but receives data via a flickering barcode on the PC screen. There is no communication back from this reader to the PC, so the user has to manually type in the response again. We do not know if the data sent using the flickering bar code is checked for authenticity before it is displayed.

[6] See http://www.zurich.ibm.com/ztic. Originally, ZTIC stood for Zurich Trusted Information Channel.

[7] http://www.agses.net

	Trusted display	Crypto	Secrets	Connected	Generic
EMV-CAP reader	✗	✓	✗	✗	✗
Gemalto's ABN-AMRO reader [1]	✗	✗	✓[a]	↔	✗
ZTIC	✓	✓	✓[b]	↔	✗
FINREAD	✓	✓	✓[b]	↔	✓
AGSES	?	✓	✓[b,c]	→	✗
Radboud Reader	✓	✗	✗	↔	✓

[a] Unknown functionality
[b] Secret key
[c] Uses fingerprints rather than PIN code

Table 1. Comparison with existing readers

Nowadays, smartcards with integrated keyboard and display are also commercially available. These displays are however very limited, for example, the smartcard offered by NagraID[8] only contains 6 characters.

A solution without even a reader is mTAN. Here the user receives an SMS on his mobile phone with transaction details and a code to confirm it. Here the phone could be seen as a trusted display. However, as mobile phones become more and more complex, they are now becoming popular targets for malware and attacks just like PCs.

6 Conclusions

The number and importance of online transactions is rapidly growing, and protecting such transactions in the face of ever more sophisticated malware is a serious challenge. This is not only an issue in online banking, but also in e-government services, or indeed any online transactions where one would really want the online equivalent of a handwritten signature.

We have presented the design of a simple and generic device for securing online transactions, which protects against any malware on the PC, including Man-in-the-Browser attacks. The device provides a trusted communication channel between a user and any remote service provider, by means of a smartcard issued by that provider.

The essence of our solution, as illustrated in Fig. 2, is quite simple: namely, make sure that all communication with the user passes through the smartcard. We are not aware of any solutions that use this approach, even though conceptually it is quite simple. This solution allows a very simple device, which does not need any cryptographic capabilities or need not store any keys or other secrets.

Because there are no secrets in the Radboud Reader, e.g. in the form of secret keys or secret protocols, anyone can make one. This can be considered a disadvantage (an attacker could make or market fake devices) but also an advantage, as anyone can implement their own device. Note that there is little incentive for manufacturers of smartcard readers to come up with solution like

[8] http://www.nagraid.com

ours, where there is no intellectual property or secret in the device, thus allowing anyone to manufacture it and not having any risk of vendor lock-in.

As discussed in detail in Section 5, the Radboud Reader offers stronger security than many existing USB-connected smartcard readers with display and keyboard used for internet banking, except IBM's ZTIC, as these solutions do not offer provide a trusted display, i.e. they can not guarantee that what appears on the display is an authentic message from the remote server.

Unlike other solutions, our solution is completely generic. The functionality hardcoded in the reader only provides some basic building blocks and the smartcard is in charge of using these to build the scenario that some service requires. So the same device can be used by different smartcards for different purposes. As the number of online services that require a high-security solution to secure transactions increases, investing in a single reader that can be used for all of them may prove a practical and economical solution. It may then also become an option to go for a slightly more expensive device, with a larger display; one could even imagine an e-reader for digitally signing electronic documents that operates in the same way as the Radboud Reader, though the limited bandwidth of communication with the smartcard could then become a bottleneck.

References

1. A. Blom, G. de Koning Gans, E. Poll, J. de Ruiter, and R. Verdult. Designed to fail: A USB-connected reader for online banking. In *Secure IT Systems (NORDSEC 2012)*, volume 7617 of *LNCS*, pages 1–16. Springer, 2012.
2. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
3. S. Drimer, S. Murdoch, and R. Anderson. Optimised to fail: Card readers for online banking. In *Financial Cryptography and Data Security*, volume 5628 of *LNCS*, pages 184–200. Springer, 2009.
4. EMVCo. EMV– Integrated Circuit Card Specifications for Payment Systems, Book 1-4, 2008. Available at http://emvco.com.
5. CEN Workshop Agreement (CWA) 14174: Financial transactional IC card reader (FINREAD), 2004.
6. Global Platform Organization. *Card Specification, Version 2.2*, March 2006. http://www.globalplaform.org.
7. P. Gullberg. Method and device for creating a digital signature, 2010. European Patent Application EP 2 166 483 A1, filed September 17, 2008, published March 24, 2010.
8. International Civil Aviation Organization. *Machine Readable Travel Documents – Part 3-2, Third Edition*. 2008.
9. ISO/IEC. ISO/IEC 7816: Identification cards – Integrated circuit cards.
10. J.-P. Szikora and P. Teuwen. Banques en ligne: à la découverte d'EMV-CAP. *MISC (Multi-System & Internet Security Cookbook)*, 56:50–62, 2011.
11. T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch. The Zurich Trusted Information Channel - an efficient defence against man-in-the-middle and malicious software attacks. In *Trusted Computing - Challenges and Applications*, volume 4968 of *LNCS*, pages 75–91. Springer, 2008.