Int. J. Inf. Secur. (2016) 15:637–657
DOI 10.1007/s10207-015-0309-6

SPECIAL ISSUE PAPER

Unpicking PLAID: a cryptographic analysis of an ISO-standards-track authentication protocol

Jean Paul Degabriele[1] · Victoria Fehr[2] · Marc Fischlin[2] · Tommaso Gagliardoni[2] · Felix Günther[2] · Giorgia Azzurra Marson[2] · Arno Mittelbach[2] · Kenneth G. Paterson[1]

Published online: 2 January 2016

Abstract The Protocol for Lightweight Authentication of Identity (PLAID) aims at secure and private authentication between a smart card and a terminal. Originally developed by a unit of the Australian Department of Human Services for physical and logical access control, PLAID has now been standardized as an Australian standard AS-5185-2010 and is currently in the fast-track standardization process for ISO/IEC 25185-1. We present a cryptographic evaluation of PLAID. As well as reporting a number of undesirable cryptographic features of the protocol, we show that the privacy properties of PLAID are significantly weaker than claimed: using a variety of techniques, we can fingerprint and then later identify cards. These techniques involve a novel application of standard statistical and data analysis techniques in cryptography. We discuss potential countermeasures to our attacks and comment on our experiences with the standardization process of PLAID.

Keywords Protocol analysis · ISO standard · PLAID · Authentication protocol · Privacy

✉ Giorgia Azzurra Marson
giorgia.marson@cased.de

Jean Paul Degabriele
jean.degabriele@rhul.ac.uk

Victoria Fehr
victoria.fehr@cased.de

Marc Fischlin
marc.fischlin@cryptoplexity.de

Tommaso Gagliardoni
tommaso.gagliardoni@cased.de

Felix Günther
guenther@cs.tu-darmstadt.de

Arno Mittelbach
arno.mittelbach@cased.de

Kenneth G. Paterson
kenny.paterson@rhul.ac.uk

[1] Information Security Group, Royal Holloway, University of London, London, UK

[2] Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany

1 Introduction

PLAID, the Protocol for Lightweight Authentication of Identity, is a contactless authentication protocol intended to be run between terminals and smartcards. The protocol was designed by Centrelink, an agency of the Australian government's Department of Human Services (DHS). According to the developers, it is supposed to provide a cryptographically strong, fast, and private protocol for physical and logical access control, without exposing "card or cardholder identifying information or any other information which is useful to an attacker" [7,19,38].

PLAID was initially proposed for use in the internal ID management of Centrelink [29]. However, the intended scope of applications has since significantly broadened to include the whole of DHS and the Australian Department of Defence [39]. Indeed, the protocol's promoters aspire to broader commercial and governmental deployment, including on an international level [13]. Strategies that are mentioned to support these aspirations include freely available intellectual property and outreach to other governmental organizations. To the latter end, NIST organized a workshop to explore the potential of PLAID for US Federal Agencies in July 2009 [32].

Another strategy that is being actively pursued is standardization. PLAID was previously registered as the Australian standard AS-5185-2010 [38] and was then entered into the ISO/IEC standardization process via the fast-track proce-

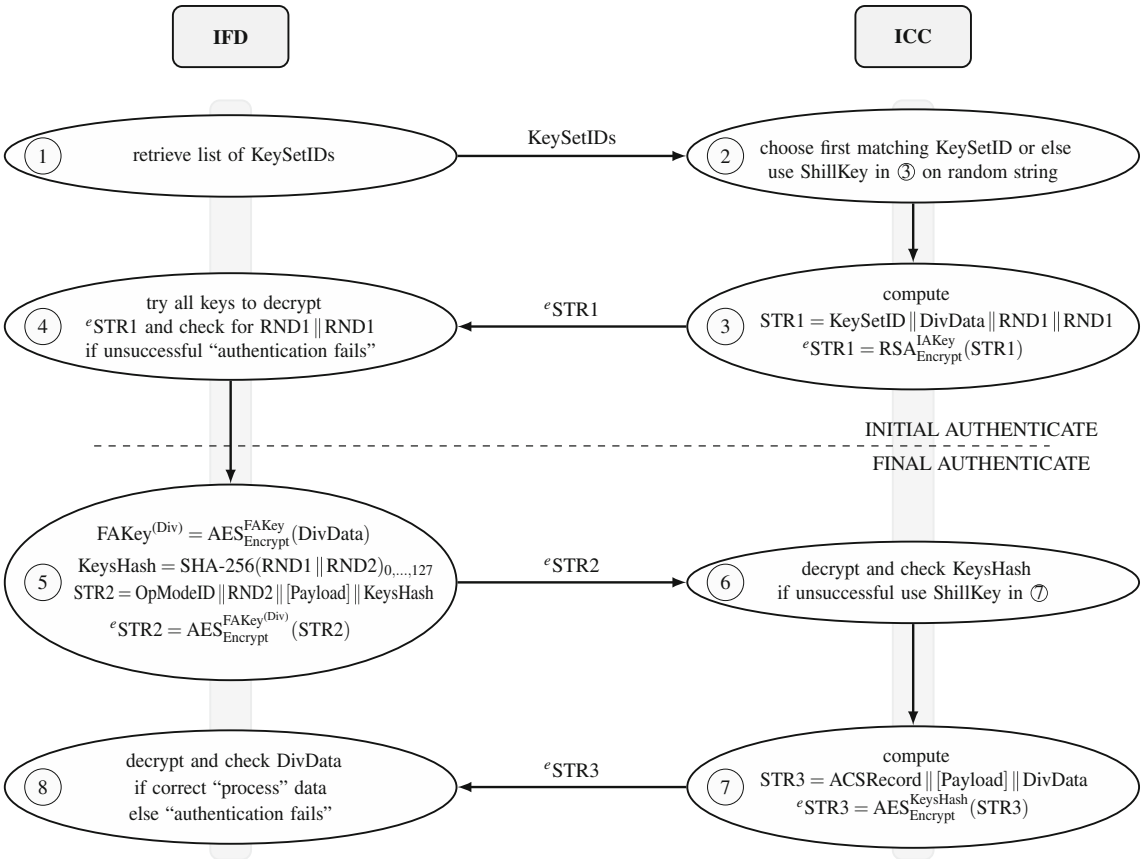

Fig. 1 PLAID protocol overview. See Sect. 2 for a detailed description of the protocol steps

dure. At the time of writing, the current ISO/IEC version is draft international standard (DIS) 25185-1.2 [19] and is currently in the "Publication stage" 60.00 (International Standard under publication). Minor changes in the original protocol to match the international standard have been applied. Reference implementations, based on the Australian standard, are available from both the Australian Department of Human Services (in Version 8.04) and the Australian Department of Defence (in draft version 1.0.0).

1.1 The protocol

The main goal of the protocol is to perform mutual authentication and establish a shared key between the terminal (IFD) and the card (ICC). To this end, the terminal and the card exchange nonces RND1 and RND2 in encrypted form and then derive the session key as part of the hash value of the two nonces. Encryption here uses both asymmetric RSA-based encryption (when the card transmits RND1 to the terminal) and symmetric AES-based encryption (when the terminal sends RND2 to the card). Authentication of the partner is presumably guaranteed by the fact that a party should know the secret key in order to be able to decrypt the other party's nonce. An overview of the protocol is depicted in Fig. 1, where the encrypted nonces are exchanged with transmissions $^eSTR1$ and $^eSTR2$. The card confirms the receipt of RND2 by sending a string encrypted under the derived key in $^eSTR3$.

The role of the terminal's initial message KeySetIDs is as follows. Each PLAID deployment involves a set of key pairs consisting of an RSA key and an AES key. Each terminal and each card stores a certain subset of these pairs. More precisely, each terminal holds a set of RSA key pairs (both encryption and decryption key) and corresponding AES master keys, while each card holds a set of RSA public keys and card-specific AES keys, derived from the corresponding AES master keys using a card identity. The keys held by a card are intended to control what types of access the card should have, so each key represents a capability. The actually deployed pair of keys is negotiated during the protocol itself, by having the terminal send a sequence of supported RSA key identifiers KeySetID in the first message. Even though the encryption key in RSA is usually public, in PLAID it is kept secret to enhance privacy (since, for example, the set of RSA keys held by a card could be used to identify the card and track its use in a deployment).

One distinctive feature of the PLAID protocol is that the card switches to using a pair of the so-called shill keys in

case of an error. That is, if the card detects some potential error, then it uses its card-specific RSA shill key and AES shill key to encrypt random data. This mechanism is intended to hide information about failures from an adversary and thereby prevent leakage about which keys are possessed by a particular card.

1.2 Previous security analyses

Centrelink's accompanying description of PLAID [7] claims that PLAID is highly resistant against leakage of card or cardholder identifying information, against various forms of active attacks, and provides mutual authentication. The document states as a goal that the protocol shall be "evaluated by the most respected cryptographic organisations, and the broader cryptographic community." For version 8, the document [7] refers to the input by various agencies like NIST and of "a number of independent cryptographic experts and consultants, a number of respected commercial cryptographic teams, as well as the internal Centrelink team."

However, we are not aware of any publicly available cryptographic evaluation of PLAID. None of the claimed security properties is backed up by arguments, nor matched against more precise formalizations in the description [7] or standards [19,38]. Some useful comments about the protocol's security have been given by the national representatives on the first DIS version of the ISO standard [18] during the disposition of comments [21]. These comments refer partly to the points discussed in Sect. 5, where we assess them in a cryptographic context.

PLAID has been scrutinized to some extent by using formal methods and automated tools. Watanabe [41], using Scyther, and Sakurada [36], using ProVerif, confirmed that PLAID satisfies some form of mutual authentication and some level of secrecy of the session key, assuming idealized cryptographic primitives. It remained unclear to us what this assurance means in a cryptographic sense. Neither of the works considers privacy aspects.

Finally, the Master's thesis of Kiat and Run [28] at the Naval Postgraduate School compares PLAID with a similar protocol, the ANSI/INCITS 504-1-2013 standard OPACITY. The conclusion is indecisive and is primarily based on deployment characteristics. The authors evaluate cryptographic properties only on a superficial level. Indeed, while the thesis does not pinpoint at any major weakness in OPACITY, a cryptographic analysis [9] was less positive.

1.3 Our results

According to the developers of PLAID, the lack of privacy in previous efforts was one of the main reasons to introduce a new authentication protocol [35]. Indeed, PLAID is described as highly resistant against "the leakage of individually identifiable, unique or determinable data or characteristics of the smart card or the holder during authentication." [7]. We argue here that PLAID does not achieve this ambitious goal. More precisely, we describe and evaluate a suite of attacks that break the privacy goals of PLAID, enabling cards to be efficiently identified in a number of realistic scenarios. We also identify some countermeasures to our attacks.

In more detail, our first attack (which further divides into three sub-scenarios) exploits PLAID's use of shill keys, which, being card-specific, can serve as a proxy for the card identity. While the shill keys themselves are not transmitted in the protocol, we show how they can be statistically estimated from RSA ciphertexts observed in protocol runs, enabling cards to be first fingerprinted and then later re-identified. This "shill-key fingerprinting" attack, presented in Sect. 3, deploys different techniques to perform the statistical estimation in three distinct attack scenarios. For two scenarios, our attack uses the standard solution to what is known as the "German Tank Problem," which concerns estimating the maximum of a discrete uniform distribution from a number of samples, while, for the third scenario, it uses clustering techniques (and in particular the standard k-means clustering algorithm) to perform the estimation of the shill keys.

Our second attack, targeting the terminal's initial message KeySetIDs, is called "keyset fingerprinting" and is presented in Sect. 4. It exploits specific properties of the protocol flow to extract information about the set of keys held by a given card, potentially allowing us to draw conclusions about the cardholder (e.g., via access authorizations). We show that this information can be efficiently extracted by interacting with a card a number of times and observing how the protocol proceeds (or fails to proceed). The information obtained in this attack may already be sufficient to identify individual cards from among a population, depending on the exact characteristics of a given deployment. The attack can also be combined with all three variants of our first attack to increase their efficiency (by reducing the number of possible keys that need to be considered in the re-identification phase).

In Sect. 5, we make a number of other observations on cryptographic aspects of the PLAID protocol, focusing in particular on its lack of forward security, the use of weak RSA encryption, the lack of integrity protection for the symmetric encryption and a number of imprecisions in its specification. Some of the issues have already been briefly touched upon in the national body comments [21] on the previous ISO standard version [18]; some aspects, like the lack of forward security, are new.

1.4 Interaction with the responsible authorities

We promptly communicated our findings to the ISO 25185-1 project editor and a contact person at the Department of Human Services. We report and comment on their responses

to our technical results in Sect. 6. In Sect. 7, we then take a step back and look at the standardization process to which PLAID was subjected, which we find interesting in its own right, and offer our own views and reflections on it.

1.5 Note

A preliminary version of this paper appears in the proceedings of the 1st International Conference on Research in Security Standardisation (SSR 2014), December 2014 [11]. This is an extended version. In particular, more simulation results for our original shill-key fingerprinting attacks, a statistical analysis in Sect. 3.1, the lunchtime attack of Sect. 3.3, the report on our interaction with the ISO representatives of Sect. 6, and our reflections on the standardization process of PLAID of Sect. 7 are new to this version of the paper.

2 PLAID protocol description

In this section, we give a detailed description of PLAID according to the specification of the draft ISO/IEC DIS 25185-1.2 [19]. A more concise overview of the protocol flow is depicted in Fig. 1. To make our description as close as possible to the original specification [19], we denote terminal and card by IFD (interface device) and ICC (integrated circuit card), respectively. Table 1 provides a summary of the most important fields and objects occurring in the protocol.

2.1 PLAID setup

In the setup phase, PLAID initializes both terminals (IFDs) and cards (ICCs). PLAID supports up to 2^{16} key sets, each consisting of an RSA key pair IAKey_i and an AES key FAKey_i. Each terminal and each card hold a subset of these overall possible key pairs, according to some access-control policy. However, the card only holds the public key part of the IAKey as well as a processed version of the original FAKey. More concretely, the card does not keep the FAKeys directly, but only a diversified version $\text{FAKey}^{(\text{Div})} = \text{AES}^{\text{FAKey}}_{\text{Encrypt}}(\text{DivData})$, where DivData is a 128-bit card identifier. The standard [19] highlights that these diversification data should be "random or unique." Using the diversified key instead of FAKey should retain security for other cards, in the case of a card being compromised, and hence, some of the (diversified) keys are disclosed. In addition to the RSA keys, $\text{FAKey}^{(\text{Div})}$ and the value DivData, each card receives a pair of individual distress-keys (called ShillKey): a random RSA encryption key and a random AES key. These "shill keys" should be used to encrypt random data in case an error is detected, thus camouflaging errors or de-facto aborts on the card.

2.2 Initial authenticate

The IA phase aims at exchanging the necessary information to compute the symmetric keys used in the FA phase as well as transferring DivData, the card-specific data later needed to guarante authenticity of the final message, securely to the terminal.

Step 1 (IFD)—IA Command: The interaction is initiated by the IFD, which transmits the complete sequence of supported KeySetIDs (in order of preference) to the ICC.
Step 2 (ICC)—IA Command Evaluation: Upon receiving a set of KeySetIDs, the ICC traverses the entire list of indices to find the first KeySetID it supports, which determines the IAKey for RSA encryption. To prevent timing attacks, it does not abort the search, even if a match has occurred. If no match is found, in Step 3 the ICC will encrypt a randomly generated string using its shill key. [1]
Step 3 (ICC)—IA Response: The ICC generates RND1, retrieves its DivData, and derives string STR1, together with an encryption of it under IAKey, as follows:

$$\text{STR1} = \text{KeySetID} \,\|\, \text{DivData} \,\|\, \text{RND1} \,\|\, \text{RND1},$$
$$^{e}\text{STR1} = \text{RSA}^{\text{IAKey}}_{\text{Encrypt}}(\text{STR1}).$$

The encrypted string ^{e}STR1 is sent to the IFD. Here PKCS#1 v1.5 padding [26] is used.
Step 4 (IFD)—IA Response Evaluation: The IFD trial-decrypts ^{e}STR1 with all possible private IAKeys indexed by its KeySetID list, and for each valid decryption, it checks whether the last two 16-byte blocks are equal. Again, to prevent timing attacks the IFD will continue the search even if a matching string has already been found. The (first) match is then used to extract KeySetID[2], DivData, and RND1. If no plaintext is of the anticipated format, authentication fails. [3]

2.3 Final authenticate

The FA phase permits to specify the operation mode and to exchange data, like a PIN or biometrics, needed to complete the authentication. Here the diversified key $\text{FAKey}^{(\text{Div})}$ (stored on the card and previously computed by the terminal during the IA phase) and a derived session key are used to secure the communication. The card authenticates by proving its ability to decrypt ^{e}STR2 as well as to include the

[1] The standard neither specifies the exact format nor the length of this randomly generated string.

[2] The standard is ambiguous in whether the trial KeySetID of the IFD or the value contained in ^{e}STR1 is stored.

[3] The standard does not specify what is meant by "authentication fails." We assume the protocol aborts in this case.

Table 1 Most important fields and identifiers of PLAID

Variable	Description
ACSRecord	An access-control system record for each operation mode required for authentication
DivData	A "random or unique" 16-byte ICC identifier
FAKey	A 16-byte AES key which can be seen as master key to compute the diversified key used in the protocol (only known to the IFD)
$\text{FAKey}^{(\text{Div})}$	A 16-byte AES key derived from the FAKey and used in the FA phase
IAKey	A 2048-bit pre-shared RSA key pair used in the IA phase. The ICC only knows the public key part
KeySetID	A 2-byte index value identifying an IAKey and FAKey or $\text{FAKey}^{(\text{Div})}$, respectively
OpModeID	A 2-byte index value identifying the operation mode. This value indicates which ACSRecord and payload the ICC needs to provide for authentication
RNDi	A 16-byte random string for $i = 1, 2$
KeysHash[a]	A 16-byte session key computed by IFD and ICC used in the FA phase
ShillKey	A pair of 2048-bit RSA public key and 16-byte AES key of the ICC (randomly chosen per ICC during setup). These keys are to be used instead of error messages to simulate the next step of the protocol camouflaging that something went wrong

[a] Note that in the original draft KeysHash refers to the entire 32 byte output of SHA-256(RND1 ‖ RND2) and the term *session key* is used to refer to the first 16 bytes which are used as secret key in the final message. For simplicity, we refer to the session key as KeysHash in this paper

correct DivData (transmitted in the previous IA phase) in the final message ${}^e\text{STR3}$.

Step 5 (IFD)—FA Command: The IFD generates the 16-byte nonce RND2 and computes the unique session key KeysHash (see footer of Table 1) as the first 128 bits of

$$\text{SHA-256}(\text{RND1} \,\|\, \text{RND2}).$$

Next, using the master FAKey indexed by KeySetID, it computes the diversified AES key

$$\text{FAKey}^{(\text{Div})} = \text{AES}^{\text{FAKey}}_{\text{Encrypt}}(\text{DivData}),$$

which corresponds to the AES key stored on the ICC under index KeySetID. The latter is used to encrypt

$$\text{STR2} = \text{OpModeID} \,\|\, \text{RND2} \,\|\, [\text{Payload}] \,\|\, \text{KeysHash},$$

using AES in CBC mode with the all-zero string as initialization vector, where Payload is an optional, variable-size field that depends on the operation mode. Concerning padding, the standard refers to the ISO/IEC 9797-1 method 2, where one byte `0x80` is appended, followed by blocks of `0x00` bytes until the length is a multiple of the block length. [4] The resulting string

$${}^e\text{STR2} = \text{AES}^{\text{FAKey}^{(\text{Div})}}_{\text{Encrypt}}(\text{STR2}),$$

is then transmitted to the ICC.

Step 6 (ICC)—FA Command Evaluation: The ICC decrypts ${}^e\text{STR2}$ with $\text{FAKey}^{(\text{Div})}$ and retrieves RND2. It computes the session key as described above as first half of the hash value SHA-256(RND1 ‖ RND2) and compares the result to the value KeysHash extracted from the decrypted ${}^e\text{STR2}$. If they do not match, the ICC encrypts a random byte string [5] using its AES ShillKey in the FA Response. Else Payload, if given, should be processed as specified by the implementation.

Step 7 (ICC) – FA Response: The ICC retrieves the Payload data specified by the operation mode (if necessary) and encrypts

$$\text{STR3} = \text{ACSRecord} \,\|\, [\text{Payload}] \,\|\, \text{DivData},$$

using AES in CBC mode with the all-zero string as initialization vector. Again, Payload is an optional, variable-size field which may (and usually will) differ from the Payload in Step 5. The resulting ciphertext

$${}^e\text{STR3} = \text{AES}^{\text{KeysHash}}_{\text{Encrypt}}(\text{STR3}),$$

is transmitted as final message to the IFD.

Step 8 (IFD)—FA Response Evaluation: The IFD decrypts the value ${}^e\text{STR3}$ and checks whether the recovered DivData matches the one received in the IA phase: if so, then the other data are considered authenticated and

[4] Though referring to ISO/IEC 9797-1 method 2, the PLAID draft standard explicitly describes a different padding method and thus makes unambiguous decoding impossible (cf. Sect. 5.4).

[5] Again, the standard does neither specify the exact format nor the length (note that STR3 in Step 7 contains a variable sized field Payload) of this random byte string.

processed according to the implementation; otherwise, authentication fails.

3 Shill-key fingerprinting: tracing cards in PLAID

According to the developers of PLAID, privacy was one of the main reasons to introduce a new authentication protocol. In this and the next section, we present two attacks on the privacy of PLAID contradicting the claims that no static information is available to be exploited.

In this section, we focus on the *traceability* of cards, that is, we consider an adversary who learns some information about one or more cards and then tries to identify these cards at a later time. We consider three distinct attack scenarios, each consisting of a fingerprinting phase and then an identification phase. The difference is roughly that in the first scenario the fingerprinting is a supervised learning phase in the sense that we can attribute execution traces to several cards, whereas the second setting corresponds to unsupervised learning where we get a set of random traces. In the third scenario, we focus on tracing a specific card throughout a system independent of the overall number cards. More precisely, the three settings are as follows.

- In the first scenario, we allow the adversary to first interact in turn with each and every card in the system in a number of protocol runs (the fingerprinting phase). We then draw a card at random and let the adversary interact with this specific card a number of times, with the adversary's goal being to identify which of the cards was selected. The adversary's ability to interact with each card in the system in turn in the fingerprinting phase (first phase) is not wholly realistic. However, given the high success rates of this attack that we will report below, we believe that good success rates would still be achieved in the more realistic scenario where the adversary does not have the guarantee of being able to interact with each distinct card in turn in a first phase, but instead must build up its picture of the system as it goes along.
- In the second scenario, which is much more challenging for the adversary, we do not allow the adversary to interact in turn with every card in a number of protocol runs, but simply present it with a sequence of transcripts of individual protocol executions, each execution involving a randomly chosen card. The identification phase and the adversary's goal are the same as before. This much more demanding attack scenario models a situation where the adversary cannot interact many times with each distinct card during fingerprinting, but only in one protocol run at a time with a random card.
- The third scenario focuses the attention on tracing a specific card without any knowledge of the other cards in the system or even their number. Here, the adversary is given a sequence of transcripts of protocol executions by a certain card in the fingerprinting phase. In the identification phase, the adversary is presented a second sequence of transcripts, which was produced by either the same card or a randomly generated one, and has to decide which is the case. This attack scenario captures an adversary that is interested in tracing a specific user throughout a system after being able to interact with this card initially for a certain amount of time and without knowing the total number of cards in the system.

In Sect. 4, we will consider a different type of attack and show how an adversary can learn the capabilities of a card (that is, it learns *which* keys are stored on a card). Besides being a serious breach of privacy on its own, this attack can also be combined with the attack (in all variants) described in this section to gain better performance.

Our attack in this section specifically targets the shill key values used by PLAID. A shill-key pair, generated for every card, contains an RSA public key and an AES key that are to be used in the IA and in the FA phases, respectively, in place of the actual keys should an error in the terminal message be detected. Intended as a security measure—to prevent attackers from exploiting potential information leaked by error messages—the use of the shill key turns out to drastically weaken the anonymity properties of PLAID.

Before explaining the details of the different attack variants, we note that in order to run the attack in this section, we need to be able to force cards into replying with RSA ciphertexts generated using their shill key in the first phase of the protocol. This is easily arranged by sending the card a first message containing an empty sequence of KeySetIDs, or a set of KeySetIDs containing a single and particularly high index that is not in use in any card on the system. Thus, we may assume that the adversary is able to gather samples of shill-key ciphertexts from cards at will.

3.1 Tracing cards via shill-key ciphertexts

We consider the following situation: we assume the system has t cards with corresponding shill-key moduli $N_1, \ldots, N_t$, where each N_j is an n-bit RSA modulus (the current draft version gives $n = 2048$ [19]). We start with our basic attack, tailored to the first scenario. In a first phase, the adversary learns, for every card in the system, k_1 encryptions of a random message under the card's shill key (N_j, e_j); then, in a challenge phase the adversary is given k_2 fresh ciphertexts (again for random messages) computed under shill key (N_{j^*}, e_{j^*}), for j^* chosen uniformly at random from $\{1, \ldots, t\}$. The adversary's goal is to identify from which card the challenge ciphertexts come, that is, to output the correct index j^*. We define the adversary's advantage

as its success probability bounded away from the guessing probability $\frac{1}{t}$.

The idea of our basic attack is that, although the shill keys are meant to be kept private, each of the k_1 ciphertexts $X_{i,j}$ computed using (N_j, e_j) leaks some information about the modulus N_j. Specifically, we learn that $X_{i,j} < N_j$ for each i. Similarly, in the challenge phase, where we have k_2 ciphertexts computed using (N_{j^*}, e_{j^*}), each ciphertext leaks some information about the challenge shill-key modulus. Starting from this observation, we now seek a procedure to obtain a good estimate of the shill-key moduli given only a certain number of corresponding ciphertexts for each modulus.

The problem can be reposed as follows. Notice that each ciphertext $X_{i,j}$ can be regarded as a uniformly random integer in the range $\left[1, N_j - 1\right]$. We are then faced with the task of estimating N_j, which is one more than the size of the interval from which the sample comes. This is essentially an instance of a classical statistical problem that is known as the *German Tank Problem*[6]. A naive approach would be to use twice the mean value of the samples $X_{i,j}$ as an estimate for N_j. A statistically strictly better approach is to use as an estimator for N_j the value

$$\tilde{N}_j = M_j + \frac{M_j}{k_1},$$

where M_j is the maximum value of the observed samples $X_{i,j}$ for $i = 1, \ldots, k_1$, and k_1 is the number of samples for each shill key. It basically corresponds to the maximum plus the average distance of observed samples. This estimator arises from a frequentist interpretation of the problem, and has the benefit of providing what is known as a minimum-variance unbiased estimator (MVUE). It can be replaced by a more appropriate Bayesian estimator, but the estimator above is sufficient for our purposes.

Our basic attack proceeds using this estimator as follows. In the first phase, we use it to produce estimates $\tilde{N}_j$ for each of the shill-key moduli N_j. In the challenge phase, we again use it to produce an estimate $\tilde{N}^*$ for the challenge shill-key modulus (now with parameter k_2, representing the number of samples available in that phase). We finally output as our guess for the challenge index j^* the index j for which $\tilde{N}^*$ is closest in absolute value to $\tilde{N}_j$, that is,

$$\arg\min_j \left| \tilde{N}^* - \tilde{N}_j \right|.$$

This concludes the description of our basic attack tackling the first scenario.

[6] See [24] for a good introduction. The name stems from the problem initially being posed as that of estimating the total number of tanks in the German army from observing a subset of their serial numbers.

3.1.1 Simulation results

We have conducted extensive simulations of the basic attack detailed above for various values of t (the number of cards) up to $t = 10{,}000$, k_1 (the number of ciphertext samples per card available in the first phase), and k_2 (the number of ciphertext samples in the challenge phase). Figure 2 depicts the results of our simulations for $k_1 = 100$ first-phase samples on the left and $k_1 = 1000$ samples on the right side.

It can be seen that, independently of the number of cards in the system, our attack significantly outperforms the basic probability for guessing the card's identity. To be precise, for any fixed value of k_1 and k_2, the attack's success probability exceeds the guessing probability by a constant factor. For $k_1 = 100$ samples in the fingerprinting phase, shown in the left plot of Fig. 2, $k_2 = 10$ challenge samples already suffice to surpass the *baseline* by a factor of three. This advantage increases to 15 times and 30 times the guessing probability for $k_2 = 50$ (resp. $k_2 = 100$) challenge samples, which can be obtained from a card within approximately 15 s (resp. 30 s) given the target execution time of the PLAID protocol of 300 ms. Our analysis also indicates that the maximal achievable success probability is bounded by the number of fingerprinting samples k_1 as values for k_2 higher than k_1 do not increase the attack's success probability further.

Unsurprisingly, increasing the number k_1 of samples available in the first phase of the attack improves the attack performance, as exemplified for $k_1 = 1000$ in the right plot of Fig. 2. While the success rates for k_2 values of 10, 50, and 100 are very close to those for the same k_2 values with $k_1 = 100$ fingerprinting samples, higher values of k_2 make the attack perform significantly better. Given $k_1 = 1000$ for fingerprinting, the attack exceeds the guessing probability by factors of over 100 or even 300 for $k_2 = 500$ resp. $k_2 = 1000$ challenge samples and large numbers of cards t.

In terms of concrete success probabilities of our simulation, our attack can with $k_2 = 1000$ challenge samples (collectable within 5 min) identify a card among 10000 cards with approximately 3 % probability (compared to 0.01 % with guessing), among 1000 cards with approximately 25 % probability (guessing: 0.1 %), and among 100 cards with approximately 75 % probability (guessing: 1 %).

3.1.2 Statistical analysis

In the following, we derive theoretical predictions for the success probability of our basic attack. For a variable X, we denote by $\mathbf{E}[X]$ its expected value, and by $\mathbf{Var}[X]$ its variance. In the following analysis, we will assume that the moduli $N_1, \ldots, N_t$ are uniformly distributed over the (integer) interval $[2^{n-1} + 1, 2^n - 1]$ which, with abuse of notation, we denote henceforth by $[\frac{\Delta}{2}, \Delta]$ to ease readability. We assume, without loss of generality, that $N_1 \leq \cdots \leq$

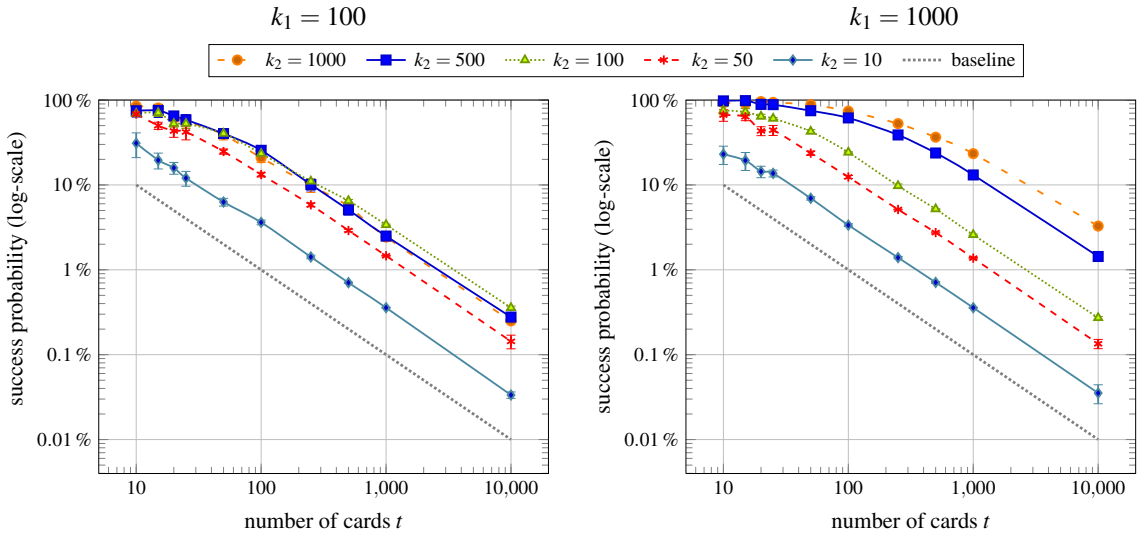

Fig. 2 Simulations of the basic shill-key attack for k_1 (the number of samples during the fingerprinting phase) equal to 100 on the *left* and 1000 on the *right*, and varying values of k_2 for each. The simulation was done with $t = 10, 15, 20, 25, 50, 100, 250, 500, 1000, 10{,}000$ cards, and the success probability is averaged over four runs with t^2 repetitions of the identification phase for all values except $t = 10{,}000$, which was averaged over four runs with 100,000 repetitions only due to computational complexity reasons. The *baseline* indicates the success probability of an adversary that tries to win the game by pure guessing. Both axes are in logarithmic scale, and *error bars* show the standard deviation

N_t. Similarly, for all $j \in \{1, \ldots, t\}$ we assume that the sample ciphertexts $\{X_{ij}\}_{i=1,\ldots,k_1}$ are uniformly distributed over $[0, N_j - 1]$, as are the ciphertexts $\{Y_i\}_{i=1,\ldots,k_2}$ over the interval $[0, N_{j^*} - 1]$. To ease the analysis, we further assume that the estimated moduli in the first phase are exact, i.e., $\tilde{N}_j = N_j$ for all j. We note that the latter assumption is equivalent to allowing for an infinite number of observations in the first phase (i.e., $k_1 = \infty$).

Let $N_{j'}$ be such that $\delta := |N_{j^*} - N_{j'}|$ is minimum among the distances $|N_{j^*} - N_j|$ for $j \neq j^*$, as illustrated below. In other words, if in our attack we would predict the wrong index, then that index will most likely be j'. This observation allows us to deduce (a lower bound for) the success probability of our attack (Fig. 3):

$$p_{\text{succ}} := \Pr[\text{attack succeeds}] \geq \Pr\left[|\tilde{N}^* - N_{j^*}| \leq \frac{\delta}{2}\right]. \tag{1}$$

We consider the German Tank estimator $\tilde{N}^*$ for the targeted modulus N_{j^*}:

$$\tilde{N}^* = M + \frac{M}{k_2}, \quad \text{where } M = \max_{i=1,\ldots,k_2} Y_i,$$

Equation (1) yields in this case:

$$\begin{aligned} p_{\text{succ}} &\geq \Pr\left[N_{j^*} - \frac{\delta}{2} \leq M\left(1 + \frac{1}{k_2}\right) \leq N_{j^*} + \frac{\delta}{2}\right] \\ &= \Pr\left[z^- \leq M \leq z^+\right] = F_M(z^+) - F_M(z^-) \end{aligned}$$

where $z^{\pm} = \frac{k_2}{k_2+1} \cdot (N_{j^*} \pm \frac{\delta}{2})$ and F_M is the cumulative distribution function of the maximum sample M. More explicitly,

$$F_M(x) = \Pr[M \leq x] = \begin{cases} 0 & \text{if } x < 0 \\ \left(\frac{x}{N_{j^*}}\right)^{k_2} & \text{if } 0 \leq x \leq N_{j^*} \\ 1 & \text{otherwise} \end{cases}$$

A good approximation of δ is given by the average distance between any two adjacent moduli. Since they are uniformly distributed in $[\frac{\Delta}{2}, \Delta]$, we have $\mathbf{E}\left[N_{j^*}\right] = 3\Delta/4$, and for all $j = 1, \ldots, t-1$ we have $\mathbf{E}\left[|N_{j+1} - N_j|\right] = \Delta/2t$. So, in particular, we use the approximations $\delta \approx \Delta/2t$, and $N_{j^*} \approx 3\Delta/4$. A direct calculation using these values produces:

$$p_{\text{succ}} \approx \begin{cases} \left(\frac{k_2}{3t} \cdot \frac{3t+1}{k_2+1}\right)^{k_2} - \left(\frac{k_2}{3t} \cdot \frac{3t-1}{k_2+1}\right)^{k_2} & \text{if } k_2 \leq 3t \\ 1 - \left(\frac{k_2}{3t} \cdot \frac{3t-1}{k_2+1}\right)^{k_2} & \text{otherwise} \end{cases} \tag{2}$$

A graphical representation of the estimated probability functions p_{succ}, for fixed $k_2 = 100, 1000$ and variable t, is given in Fig. 4. In the same figure, we compare the theoretical predictions with the corresponding simulation results. Notice that the observed success rates and their theoretical predictions are close and coincide asymptotically. Also note that p_{succ} provides a lower bound for the attack's success probability under the assumption that the moduli $N_1, \ldots, N_t$ are known, or equivalently that $k_1 = \infty$. In fact, within our simulations—and in any realistic scenario—the attack relies

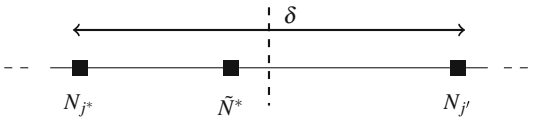

Fig. 3 Results of the statistical analysis for the basic shill-key fingerprinting scenario: the predicted success probability of the attack refers to $k_1 = 1000$, $k_2 = 100$ (*loosely dashed*) and $k_2 = 1000$ (*densely dashed*). The *baseline* indicates the success probability of an adversary that tries to win the game by pure guessing. The number of cards t on both sides and the success probability on the right side are displayed in logarithmic scale; *error bars* show the standard deviation

on the first-phase estimates $\tilde{N}_1, \ldots, \tilde{N}_t$ rather than on the exact moduli $N_1, \ldots, N_t$. We thus expect the actual success probabilities to be smaller than the respective predictions. This behavior is indeed visible in Fig. 4.

3.2 Tracing cards from a mixed set of shill-key ciphertexts

For the first scenario and our basic attack in Sect. 3.1, we assumed that during the initial phase the attacker was able to identify ciphertexts computed from the same key. In our second scenario, we relax this assumption: we now give the attacker a large mixed set of $k_1 \times t$ ciphertext samples, each sample coming from a randomly selected card. The challenge phase of the attack proceeds as before where the attacker obtains a small sample of k_2 ciphertexts computed by the same card, and the attacker's goal is to identify this card.

The challenge now is to somehow process this mixed set of samples in order to extract reasonable estimates of the individual RSA moduli. We accomplish this by means of a heuristic clustering technique. Assuming that we know the number of cards t used to produce the mixed sample set, let $N_1, \ldots, N_t$ represent their shill-key moduli in increasing order. From the mixed sample of ciphertexts, we ignore all samples smaller than 2^{2047}. We then use a standard clustering technique based on the k-means algorithm to group the remaining ciphertext samples into t clusters approximating the intervals $[N_j, N_{j+1})$, for $j \in \{0, 1, \ldots, t\}$ and $N_0 = 2^{2047}$. Once we have this set of clusters, we then obtain an estimate for the shill-key modulus N_{i+1} by using the German Tank estimator on the cluster corresponding to the interval $[N_i, N_{i+1})$.

We now describe this clustering attack for the second scenario in more detail. We initially assign to each of the t clusters a uniformly random value in the range $(2^{2047}, 2^{2048})$. This value is called the *centroid* of the cluster. For each ciphertext sample greater than 2^{2047}, we calculate its distance from each of the cluster centroids and assign that ciphertext to the cluster to whose centroid it is closest. The distance metric is merely the absolute value of the arithmetic difference. Once that every ciphertext sample has been assigned to a cluster, we ensure that no cluster is empty. If an empty cluster is found, we pick another cluster at random whose size is greater than one and move its largest element to the empty cluster. We then set the centroid of each cluster to be the mean of the ciphertext samples contained in that cluster, as per the standard k-means algorithm. We iterate this process of assigning ciphertext samples to clusters and recalculating their centroids until the centroids converge to stable values, or the maximum number of iterations is exceeded.

In the challenge phase, the attacker is given k_2 ciphertexts which are all computed by the same card. Here, our clustering attack proceeds identically to the previous one: the attacker

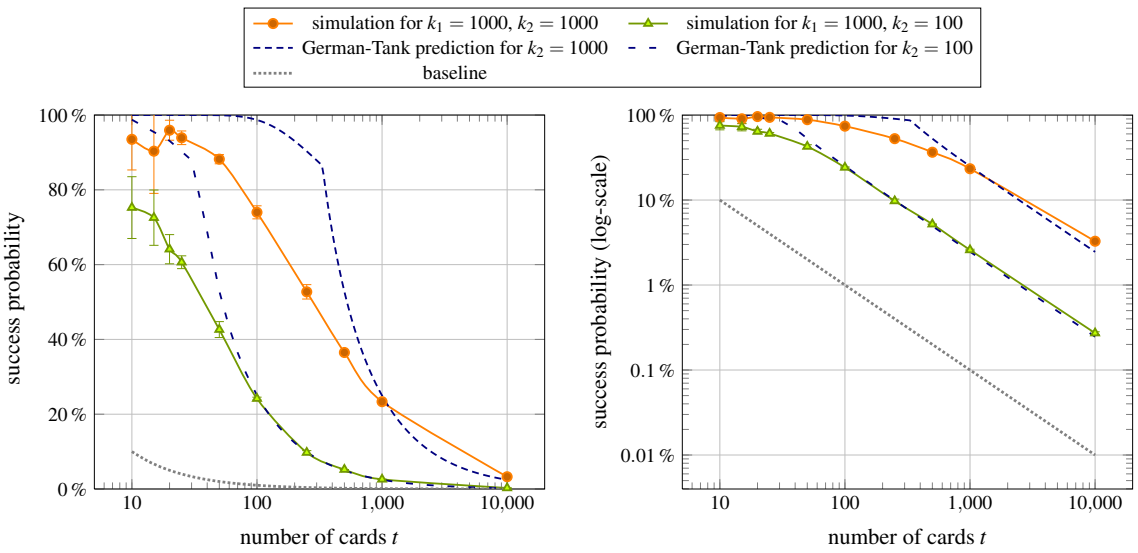

Fig. 4 A pictorial explanation of Eq. (1): if N_{j*} is the target modulus, $N_{j'}$ its closest modulus, and $\tilde{N}^*$ our estimate of N_{j*}, then our attack succeeds whenever $\tilde{N}^*$ is at most $|N_{j*} - N_{j'}|/2$-close to N_{j*}

uses the estimator to produce an estimate $\tilde{N}^*$ for the challenge shill-key modulus and outputs as its guess the index of the modulus from the first phase that is closest to $\tilde{N}^*$.

3.2.1 Simulation results

We ran simulations of the above clustering attack for a mixed sample set of size $t \times k_1$ for various values of t (the number of cards) up to $t = 100$, values of k_1 first-phase samples equal to 100 and 1000, and different numbers k_2 of second-phase samples. Figure 5 depicts the results of our simulations for $k_1 = 100$ on the left and $k_1 = 1000$ samples on the right side.

Working in a more ambitious scenario, the success probabilities for the clustering attack are considerably lower than for the basic attack in the first scenario. In particular, it does not significantly benefit from higher numbers k_2 of challenge-phase samples like the previous attack. Even for $k_2 = 1000$ challenge ciphertexts the success rate for $t = 100$ cards stays around 5 % (independent of k_1), while the basic attack (cf. Fig. 2) successfully identified the challenge card with probability 20 % (for $k_1 = 100$) or even 74 % (for $k_1 = 1000$). Nonetheless, we see that our clustering approach is able to correctly identify a card with a significant advantage over the guessing probability, i.e., with a success probability approximately four times higher than the latter.

Notably, as we increase the number k_1 of first-phase samples to 1000, we do not get a corresponding increase in performance as in the previous attack but rather obtain roughly the same success probabilities as for $k_1 = 100$. On the other hand, for low parameter values k_2 the clustering attack in both cases performs comparable to the basic attack. In fact, if we compare Figs. 2 and 5, we see that for $k_2 = 10$ the two attacks achieve almost identical success rates. Therefore, if the attacker is limited to a small number of samples ($\approx$10) during the identification phase, he can trace cards as effectively using the clustering attack without requiring a sorted set of ciphertext samples during fingerprinting.

3.3 Tracing cards during lunchtime

We now turn to the third scenario which models an attacker that tries to trace a single card throughout a system with an unknown total number of cards, after being able to interact with the target card for a certain (short) amount of time (e.g., during lunch) at the beginning of the experiment. In contrast to the previous two scenarios involving many keys to be re-identified and where the attacker was able to sample ciphertexts from all cards in the experiment's first phase, this third scenario now poses a distinguishing challenge between a target card from which the attacker is able to sample ciphertexts up-front and a second, random card (which it did not have access to before).

More precisely, in the fingerprinting phase the attacker is given a set of k_1 ciphertext samples under a fixed, but randomly generated shill key. The challenge phase proceeds by providing the attacker with a sample of k_2 ciphertexts generated either by the same shill key as in the first phase, or by a second, randomly generated shill key. The attacker's goal is to decide whether it is interacting with the same card as before or a new card, and we define its advantage to be its suc-

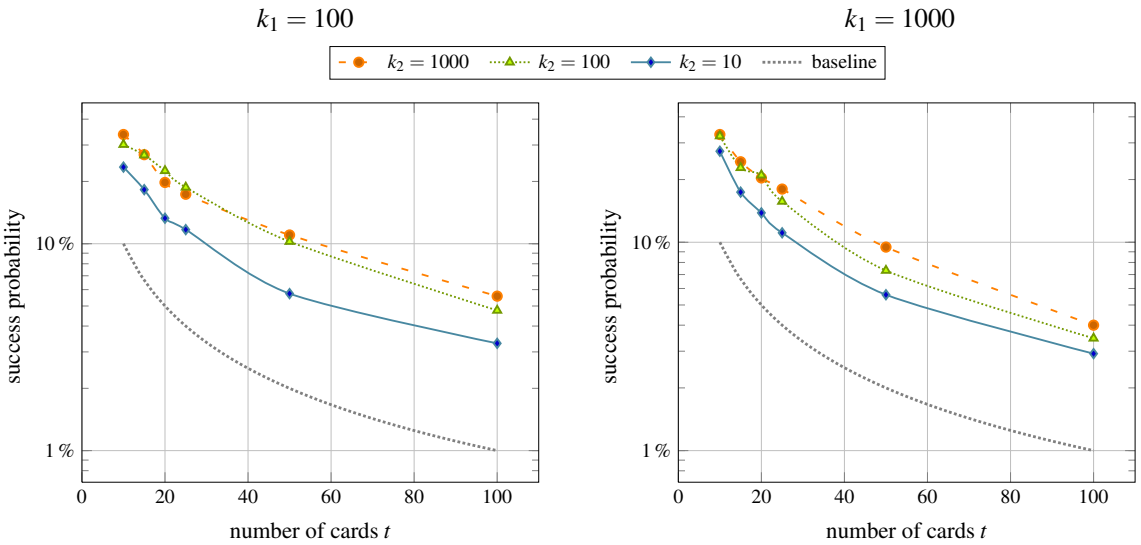

Fig. 5 Simulations of the clustering attack for k_1 (the number of samples during the fingerprinting phase) equal to 100 on the left and 1000 on the right, and varying values of k_2 for each. The success probability, shown in log-scale, is averaged over fifty runs, and the simulation was done with $t = 10, 15, 20, 25, 50, 100$ cards. The *baseline* indicates the success probability of an adversary that tries to win the game by pure guessing

cess probability in doing so, suitably adjusted by subtracting the guessing probability $\frac{1}{2}$.

While this third scenario might at first glance seem strictly weaker than our first scenario, it is actually not, since the first scenario allows the attacker to sample ciphertexts from *all* cards, while the third scenario is about distinguishing a "known" card from an "unknown" one. This setting is indeed quite realistic. First of all, it does not make the assumption that all cards, as in the previous scenarios, can be sampled up-front—which might indeed be a hard task for many types of attackers. Secondly, tracing a single card and thereby, e.g., the movements of a particular person within a certain area where the attacker is able to set up fake PLAID terminals constitute a realistic threat scenario for a PLAID deployment.

Coming to our lunchtime attack tailored to this third scenario, the strategy to decide in the second phase whether the attack faces the previously sampled card or a new one is based on the simple idea that if the ciphertexts in the second phase come from the same card, then an estimate for the modulus in the second phase should be close to the estimate in the first phase. The question then becomes how to define "close" so as to produce an effective attack. For this, we note that the variance of the MVUE estimator,

$$\mathbf{Var}\left[\tilde{N}\right] = \frac{1}{k} \cdot \frac{(N-k)(N+1)}{k+2},$$

only depends on the estimate of the target card's modulus $\tilde{N}$ and the number of samples k. We then define "close" according to the "3-sigma rule of thumb", i.e., we say that the samples come from the original card if the absolute difference between the two estimates falls within three times the standard deviation,

$$3\sigma = 3 \cdot \sqrt{\mathbf{Var}\left[\tilde{N}\right]}$$

where the variance is computed relative to $k = \min(k_1, k_2)$ and the estimated modulus $\tilde{N}$ of the target card. This means that the probability that an estimate produced using ciphertexts sampled from the original card lies within the confidence interval is about 98 %. While giving us a false reject rate (FRR) of 2 % choosing 3σ as the confidence interval also provides us with an estimate for the false accept rate (FAR), i.e., the likelihood of a random card being classified as the actual target card. False positives occur with large probability if the modulus of the random card is within distance of the confidence interval (3σ) from the modulus of the target card. For $k = 100$, the confidence interval ranges over about 5 % of the modulus space:

$$\frac{3\sigma}{2^{2047}} \approx 5\,\% \quad \begin{array}{l}\text{for the mean modulus } N = 2^{2048}-1-2^{2046} \\ \text{and } k = 100\end{array}$$

Hence, for $k = 100$ we should expect an FAR increase in about 5 %. This value drops linearly with higher values of k, that is for $k = 1000$ we only get an increase in the FAR of about 0.5 %.

3.4 Simulation results

We ran simulations of the lunchtime attack described above for various values of k_1 (the number of first-phase samples)

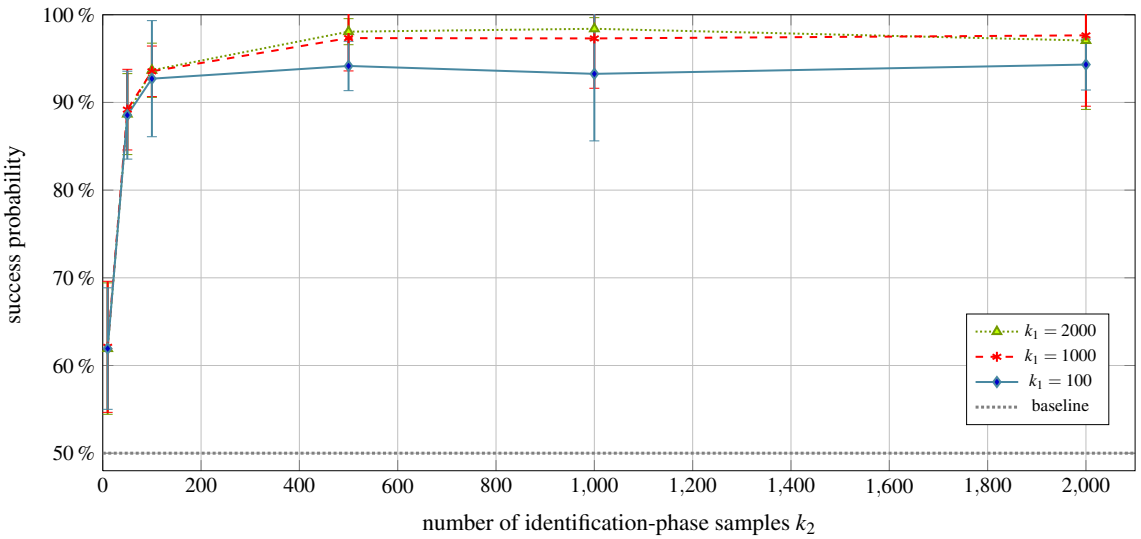

Fig. 6 Simulations of the lunchtime attack for different numbers k_1 of samples during the fingerprinting phase and varying numbers of identification-phase samples k_2. The success probability is averaged over 100 runs with 100 challenge phases each. The *baseline* indicates the success probability of an adversary that tries to win the game by pure guessing. *Error bars* show the standard deviation

and k_2 (the number of second-phase samples)[7]; Fig. 6 depicts the results.

As expected, the success rates of our lunchtime attack are quite high. (But note that they are formally incomparable to the simulation results in previous attack scenarios as the attacker here is only challenged with making a binary decision.) Notably, even with small numbers of fingerprinting and identification samples such as $k_1 = 100$ and $k_2 = 50$ (which can be obtained from a card within approximately 30 s for fingerprinting and 15 s for identification given the target execution time of the PLAID protocol of 300 ms), our lunchtime attack already shows a success rate of 90 % (the guessing probability is 50 %). If the attacker is given more time in the second phase, the probability of correctly distinguishing the initial from a random card stabilizes above 90 % for values $k_2 > 100$.

We moreover observe the same pattern seen already in the basic attack (cf. Fig. 2) with respect to the ratio between k_1 and k_2: obtaining more identification-phase samples (k_2) than fingerprinting-phase samples (k_1) seems not to help the lunchtime attack to further increase its success probability. In more detail, the success probabilities for $k_1 = 100$, $k_1 = 1000$, and $k_1 = 2000$ settle at approximately 94, 97, resp. 98 % from $k_2 \approx k_1$ on. The seen success rates thus nicely fit into our theoretical predictions of having an overall error between 2 and 7 % stemming from our choice of fixing the confidence interval to a size of 3σ giving us an FRR of about 2 % and depending on the value of k an FAR of between 0.25 % (for $k = 2000$) and 5 % (for $k = 100$). The upper bound for the achievable success probability of $1 - \text{FAR} - \text{FRR}$ for our lunchtime attack is also reflected in the standard deviation (shown by the error bars in Fig. 6) of our experimental results. In order to further increase the success rate, one should thus aim for minimizing the overall error rate $\text{FAR} + \text{FRR}$ noting that an increase in the confidence interval (e.g., to 6σ) decreases the false reject rate (FRR) while increasing the FAR, and an increase in the number of samples k decreases both the FAR and the FRR but, of course, comes at the price of needing extra time to mount the attack.

3.5 Connection to key privacy of RSA encryption

We remark that our shill-key fingerprinting attack only considers properties of RSA moduli and is, thus, of independent interest in the study of *key privacy* (or *key anonymity*) of RSA encryption, a security notion introduced by Bellare et al. in [3]. In the key privacy security model of [3], an adversary plays against two key pairs and is given both the public keys. Security is modeled in terms of key indistinguishability, requiring that it is infeasible for any efficient adversary, that can request encryptions of messages of its choice under one of the two public keys, to tell which key was chosen with probability higher than guessing. As already pointed out in [3], the RSA cryptosystem does *not* provide key privacy. Security is trivially broken when the two key lengths are different. However, RSA keys of the same bit length are easy to tell apart, too: let $N_0 < N_1$ be two RSA moduli: independently of the underlying plaintext, a ciphertext c computed under one of the two corresponding keys satisfies $c < N_b$. A single-query attack which succeeds with non-negligible advantage simply requests to encrypt an arbitrary message and then compares the resulting ciphertext c with the smaller modulus: if $c < N_0$, then it returns 0 and else guesses 1.

This attack is not directly applicable to the PLAID setting because there the RSA encryption keys are kept secret. Still, our shill-key fingerprinting attack variants on PLAID can be seen as similar in spirit to, but obviously harder to perform than the above single-query attack. In particular, the third scenario (lunchtime attack) explicitly poses a distinguishing challenge (though, of course, without providing the adversary with the RSA moduli) and hence can be considered to be closest to the security notion of key privacy. Moreover, if the encryption scheme used within PLAID were to enjoy key privacy, then the attacks presented would be completely thwarted (and the public keys would no longer need to be kept secret).

3.6 Countermeasures to our attacks

A very simple countermeasure to our attacks is for every card to use the same RSA shill key. This does not seem to have any negative security consequences and renders ineffective any tracing attacks based on the analysis of RSA shill-key ciphertexts.

A second countermeasure to our attacks is to modify the RSA encryption scheme so that it is key private. This can be done in two ways: padding by adding multiples of the modulus to the ciphertext, and selection of RSA moduli that all lie in a small interval.

The first approach for the second countermeasure simply adds a random multiple kN_j of the modulus to the RSA ciphertext. Here, the range from which k is selected is chosen to make all the resulting ciphertexts for the different RSA moduli (approximately) uniformly distributed in the same interval. The larger k is, the better the key privacy is attained, but the larger is the bandwidth needed to transmit ciphertexts.

The second approach is to choose the RSA moduli for shill keys to lie in a much smaller interval than $\left[2^{n-1}, 2^n - 1\right]$. For example, one could generate the prime factors of the moduli N_j in such a way that each N_j begins with an MSB of 1 followed by 256 0-bits, i.e. so that the moduli lie in the range

[7] Note that, in contrast to the first two scenarios, the third scenario and our according lunchtime attack is independent of the overall number of cards in the system.

$[2^{n-1}, 2^{n-1} + 2^{n-256}]$. This would ensure that the distributions of RSA ciphertexts are much harder to distinguish using our methods and, heuristically, does not affect the security of the RSA encryption scheme (since there are still many RSA moduli that can be generated in this range, and there are no known speedups for factoring algorithms for RSA moduli in such a range).

Detailed analysis of the specifics of these countermeasures is beyond the scope of the current work.

4 Keyset fingerprinting: determining a card's capabilities

In this section, we present another type of attack on PLAID's privacy, which we call *keyset fingerprinting*. This attack reveals the exact set of keys a card knows[8], thereby determining its *capabilities* in terms of which keyset it can use, i.e., which specific terminal it is able to talk to. In order to mount the attack, we exploit the following observations: (i) the KeySetIDs list sent by the terminal (in the clear) in the IA Command contains all keys known by the terminal [19, Section 6.1], and is *not* authenticated [9] , (ii) in its IA Response, the card is required to use the first key of the received KeySetIDs list it knows [19, Section 6.2], and (iii) if the card uses its ShillKey, in the IA Response, then the terminal aborts [19, Section 6.4].

4.1 The attack in a nutshell

We first explain the core idea of our attack by describing a concrete attack scenario. Assume an adversary observes a successful protocol run between a card and a terminal where the latter had sent (in the clear) KeySetIDs $= (2, 5, 8)$. From this, the attacker not only learns that the ICC holds at least one of the keys with IDs $\{2, 5, 8\}$, but it can also determine *all* of the keys the ICC supports, independently of the identifiers announced in KeySetIDs. To this end, the adversary can trigger a protocol run and mount a man-in-the-middle attack as described below.

In a first phase, the attacker sequentially replaces the IFD's original initial message by one containing only a single identifier from the original list of KeySetIDs, that is, 2, 5 or 8 in our example; by observing the subsequent protocol run, the attacker deduces that the ICC supports the selected key if and only if the protocol execution reaches the third step, i.e., if the terminal responds with a third message. In a second phase, the attacker sequentially prepends to the IFD's original initial message all key identifiers that were not contained in KeySetIDs, e.g., (1, 2, 5, 8), (3, 2, 5, 8), …, (65536, 2, 5, 8) in our example. Then, from each of the subsequent protocol runs, the attacker learns that the ICC knows the inserted key if and only if the IFD does *not* respond with a third message. This is because of observation (ii) above about the first matching key in the list to be used. At the end of the two phases, the attacker knows the identifiers of all keys supported by the ICC.

We stress the attack above can be performed in a remote fashion (in the sense that the card and the reader can be far from each other) where two attackers, placed in physical proximity to the terminal, respectively, the card relay the exchanged messages between each other, playing the role of a card resp. a terminal. Moreover, this attack can be mounted independently of the values announced in KeySetIDs, as long as the attacker observes a single, successful protocol execution.

Note that knowledge of all the keys supported by a card also reveals its capabilities (e.g., access authorizations), thereby potentially disclosing highly sensitive information. While this is not, in general, sufficient to identify a card uniquely, it effectively allows to derive capability classes, containing cards with the same capabilities. Moreover, in certain scenarios, capabilities like access authorizations might even leak the identity of a card's owner, hence breaking its anonymity, as some keys might be used exclusively to access security-critical infrastructure [19, Annex C] such as server rooms or the CEO's office. The impact of keyset fingerprinting is furthermore increased by the remote nature and the low cost of the attack (in terms of the number of interactions between terminal and card). Even in large-scale, realistic scenarios, the attack requires only few seconds (and no physical proximity of card and terminal) to determine a card's capabilities. See Sect. 4.2 for a more detailed discussion.

We remark that keyset fingerprinting can, in addition, be used as a prefilter for (all variants of) our ShillKey fingerprinting attack discussed in Sect. 3. Recall that the performance of these attacks heavily depends on the number of cards in the system that have to be distinguished. By first performing keyset fingerprinting on the card(s) in question, this number can potentially be reduced substantially (thereby improving the overall efficiency), as the ShillKey fingerprinting in a second step only has to discriminate among the smaller number of cards belonging to the same capability class. Finally, we note that there are cases where the cheaper keyset fingerprinting attack on its own is actually already sufficient for a tracing attack: whenever a traced card has a *unique* set of supported keys (i.e., is the only member in its capability class), this attack is able to uniquely (re)identify that card. Furthermore,

[8] Recall that terminals announce their supported keysets by sending corresponding KeySetIDs in the clear. As a consequence, any observer can see which keys are related to which resource/terminal.

[9] We note that the unauthenticated nature of the PLAID protocol messages has already been criticized in the national body comments on an earlier ISO draft [18]. In our attack, we exploit this weakness, refuting the claim of the current ISO draft [19, Annex H.1.1] that sending KeySetIDs in clear is "of no use to an attacker."

keyset fingerprinting suffices to distinguish two cards as long as there is a key supported by only one of the cards.

4.2 The attack details

Suppose that we observe a successful authentication between an honest terminal (IFD) and an honest card (ICC). In the course of the protocol execution, the IFD starts by sending the list KeySetIDs $= (\text{KeySetID}_{i_1}, \ldots, \text{KeySetID}_{i_\ell})$ containing (all) KeySetIDs it supports. The keyset fingerprinting attack proceeds in two phases, focusing first on the keys supported by the IFD and then on the remaining keys.

> Phase 1. In the first phase, we replace the initial KeySetIDs list with a list containing only one of the keys supported by the IFD at a time, i.e., we replace the first message by (KeySetID_{i_j}) for $j = 1, \ldots, \ell$ in ℓ sequential interactions. We relay the response of the ICC unmodified to the IFD. If the IFD replies with a third message in the jth interaction, we can infer that the ICC knows the key with KeySetID_{i_j}. Otherwise, the ICC did not support this key and hence used its ShillKey, leading the IFD to abort.
> Phase 2. In the second phase, we prepend the initial KeySetIDs list with one (or multiple, see below) values $\text{KeySetID}_j \notin \{\text{KeySetID}_{i_1}, \ldots, \text{KeySetID}_{i_\ell}\}$ at a time. We relay the response of the ICC unmodified to the IFD. If the IFD replies with a third message, we can infer that the ICC knows none of the prepended keys. Otherwise, the ICC did know at least one of these keys (which the IFD does not support), leading the IFD to abort. This relies on observations (ii) and (iii) above.

We measure the attack costs in terms of the number of interactions between IFD and ICC needed to extract the keys supported by the ICC. In the first phase, which requires ℓ interactions between the IFD and the ICC, we are able to determine exactly which of the keys $\{\text{KeySetID}_{i_1}, \ldots, \text{KeySetID}_{i_\ell}\}$ supported by the IFD the ICC knows. The second phase aims at determining which of the remaining $2^{16} - \ell$ KeySetIDs are known by the ICC. There are different strategies to proceed in Phase 2:

1. The *basic approach* is to simply prepend each one of the $2^{16} - \ell$ KeySetIDs not supported by the IFD one at a time, resulting in $2^{16} - \ell$ interactions in order to determine exactly which of the keys the ICC knows. Together with the first phase, this approach leads to 2^{16} interactions to fingerprint a card.
2. In the *binary search approach*, the set of KeySetIDs is partitioned along a binary tree with the full set of all $2^{16} - \ell$ KeySetIDs at the root, the first half of them as the left child, the second half of them as the right child, etc. In the second phase, first the root (i.e., all $2^{16} - \ell$ KeySetIDs) is prepended. If the IFD replies, the ICC knows none of these keys and we have thus completed the keyset fingerprinting for the card. Otherwise, both the left half and the right half are prepended (sequentially) and, again, if the IFD replies, then the ICC knows none of the prepended keys. This process can be repeated recursively until the IFD replies for each branch.
   Using this approach, we can quickly rule out those parts of the KeySetID space where the ICC does not know any key. More precisely, denote by n the number of keys the ICC knows in total and by ℓ' the number of keys the ICC knows among the ℓ keys supported by the IFD. Then we can upper bound the number of interactions needed to fingerprint a card by $(n - \ell') \cdot \log(2^{16}) + \ell = (n - \ell') \cdot 16 + \ell$, since a traversal of the binary search tree in order to pinpoint a single key requires at most $\log(2^{16})$ (i.e., height of the tree) additional interactions.
3. The *binary search with known maximum approach* is a further optimization which is applicable in scenarios where the highest KeySetID in the system, MaxID, is known in advance. In this case, the binary tree can be reduced to the tree having only the MaxID $- \ell$ remaining unknown KeySetIDs (instead of all $2^{16} - \ell$) as leaves. The number of interactions to fingerprint a card therefore is reduced to $(n - \ell') \cdot \lceil \log(\text{MaxID}) \rceil + \ell$.

When comparing the strategies for the second phase, in the (unlikely) worst case where the card knows all 2^{16} possible keys, the basic approach requires 2^{16} interactions, whereas the binary search approach takes approximately 2^{20} interactions. We observe however that the binary search is more efficient as long as the card holds less than 2^{12} keys (which we assume to be the case in any real scenario).

4.2.1 A practical example

For the sake of providing the reader with some estimates on a more realistic, but still large-scale example, consider a scenario where MaxID $= 5000$ keys are deployed (enough for, e.g., a large building or a small campus) and the considered terminal and card both hold $\ell = n = 10$ keys, from which $\ell' = 1$ key is known by both.

We chose these parameters in light of the targeted execution time for PLAID and the resource restrictions imposed by the terminal and card hardware. Most notably, in every execution of the PLAID protocol, the terminal has to perform ℓ RSA decryptions, which is an expensive cryptographic operation for a computationally constrained embedded device. [10]

[10] For 2048-bit RSA decryptions or signatures, [34] reports times of over 100 ms for mobile devices (without cryptographic coprocessor), while our simulations on an Intel Core i7 2.4 GHz are around 10 ms.

But since the previous ISO draft aims at an overall protocol execution time of less than 300 ms [18, p. 27], this means that ℓ cannot be too large.

With these parameters, the binary search approach would require $(10-1)\cdot 16+10=154$ interactions to fingerprint the card. Knowing the highest KeySetID MaxID, this can be further optimized to $(10-1)\cdot\log(5000)+10\approx 121$ interactions using the binary search with known maximum.

4.3 Potential countermeasures against our attack

As the keyset fingerprinting attack relies heavily on the malleability of the initial KeySetIDs message sent by the terminal, tamper-protecting this message is the obvious way to prevent this attack. One potential and immediate remedy to detect and to prevent tampering with the initial message would be to let the ICC include a hash value of the KeySetIDs value in the plaintext of STR1. The terminal could then check whether the ICC obtained the unmodified initial message by comparing the hash value that it receives with the hash of the original KeySetIDs value. However, a rigorous analysis would be required to put this idea on a profound foundation. More importantly, this modification would also essentially rely on STR1 being integrity protected, which is not the case in PLAID as we discuss in the next section.

5 Further security considerations

Here we discuss further security considerations and mainly the secrecy of the established keys which, according to the standard, can be optionally used "as a secure messaging, session or encryption key in subsequent sessions." We also point out that the design of PLAID deviates in several ways from good cryptographic practice. We observe that some of these issues have already been pointed out in the comments [21] on the previous ISO draft version [18].

5.1 Forward (in)security

Forward security [2] demands that one cannot recover session keys generated in the past, even if the long-term secrets of a party become known. In the case of PLAID, the long-term secrets correspond to the secret RSA keys and the FAKey on the terminal side, and to the public RSA keys, DivData, and the diversified keys $\mathrm{FAKey}^{(\mathrm{Div})}$ on the card side. The loss of keys of either party immediately reveals all past session keys, and also of future sessions, even if they are executed honestly between the parties and the adversary merely observes these execution traces. Furthermore, revealing a card's secrets also allows the identification, a posteriori, of traces belonging to that card and so breaches privacy in this sense.

Assume first that a terminal's long-term secrets become known to the adversary and consider the trace of an execution between this terminal with an arbitrary card: the adversary can, analogously to the genuine server, try to decrypt the ciphertext encrypting string ${}^{e}\mathrm{STR1}$ under all possible RSA private keys of the terminal, until it succeeds with one key. It then obtains DivData, hence can compute $\mathrm{FAKey}^{(\mathrm{Div})}$ by executing $\mathrm{AES}^{\mathrm{FAKey}}_{\mathrm{Encrypt}}(\mathrm{DivData})$ and then decrypt ${}^{e}\mathrm{STR2}$ sent by the honest terminal to recover the session key KeysHash.

Next, suppose that the adversary gets hold of the diversification data DivData and the diversified key $\mathrm{FAKey}^{(\mathrm{Div})}$ of a card. It can then try to decrypt ${}^{e}\mathrm{STR2}$ with this key to obtain some candidate KeysHash for the session key. The adversary can verify the validity of this candidate by checking that ${}^{e}\mathrm{STR3}$ decrypts under the candidate key to the given DivData. This way, the adversary is able to identify traces belonging to the specific card and to determine correct session keys of the card.

Most importantly, any such breach would lead to the disclosure of the payload data which may be highly sensitive (for example, a user's biometric data).

5.2 Key (in)security in the Bellare–Rogaway model

The PLAID protocol specifies the option of reusing the negotiated session key KeysHash for subsequent secure communication. We comment on possible consequences of doing so. Our starting point is the widely used Bellare–Rogaway (BR) security model [1] for key exchange protocols. This model demands that all session keys should look random to the adversary. Neglecting technical details, this is formalized by presenting the adversary either the genuine session key or an independent random key and challenging it to decide which is the case. This immediately requires of a protocol that its session keys are not themselves used in a trivial way in the key exchange steps; otherwise, the adversary can try to test the given key against a protocol execution trace. In the specific case of PLAID, the adversary can try to decrypt ${}^{e}\mathrm{STR3}$ with the given key and will recover a meaningful plaintext with overwhelming probability if and only if this key equals the genuine key KeysHash. Thus, PLAID *cannot* achieve security in the BR model.

Note that the lack of security in the BR sense does not necessarily imply that a protocol is insecure. It merely means that other models must be used to assess its security. PLAID is not unique in this respect: a prominent example of a protocol not achieving BR security is TLS up to version 1.2, leading researchers to investigate various alternative security evaluations [4,6,17,22,30]. The usage of the session key in the exchange step is often alleviated by the fact that messages in this part and in the channel protocol differ in format, e.g., if a counter value is used and incremented with each application.

This form of "domain separation," however, is not necessarily given in case of PLAID, because the subsequent channel message format has not been specified.

Interestingly, PLAID could easily avoid the problems with the session key being used in the key exchange phase. Recall that the session key KeysHash for AES (with 128 bits) is derived as the first 128 bits of the value SHA-256(RND1 ‖ RND2). Since the hash value has 256 bits, one could easily use the remaining 128 bits as the AES-128 key for the final message in the key exchange step and then switch to KeysHash as before in the channel protocol. In the original protocol, the card in some sense demonstrates knowledge of $\text{FAKey}^{(\text{Div})}$ by being able to decrypt the terminal's message and answer under the derived key. This would still be true with the proposed modification. Note however that this modification still requires a formal security treatment.

5.3 On the applicability of Bleichenbacher's attack

Recall that PLAID uses PKCS#1 v1.5 padding for RSA encryption. The accompanying protocol description [7] argues that there is no need to use OAEP padding, because "PLAID doesn't expose the modulus or any other RSA primitive" and that "there is a significant performance advantage in using PKCS#1 v1.5 padding." While we do not feel inclined to comment on the performance-related issue, the first part of the argument is debatable in light of the fact that exposure of a card's secrets does reveal the public keys. Further, our attacks in the previous sections show that some information about the moduli is revealed, and the exponent e may be fixed. We note that the comments section in the previous ISO version of PLAID [18] also asks for investigations of the possibility of mounting Bleichenbacher's attack.

Once the RSA public key is known, one can in principle mount Bleichenbacher's attack [5] on PKCS# v1.5 padding. In this attack, the adversary takes a ciphertext $c \in \mathbb{Z}_N^*$ of some unknown padded message m and "shifts" the message by multiplying c with a random $s^e \bmod N$. With sufficiently high probability, the derived "message" $sm \bmod N$ is PKCS#1 v1.5 padding compliant. The adversary could thus potentially deduce information about m in case of an error message [11] indicating correct or incorrect padding, and given sufficiently many error messages, recover m. The attack has been significantly improved in a series of papers, e.g., [23,31].

For PLAID, the message format carries some redundancy in terms of repeating RND1. Therefore, the shifted message $sm \bmod N$ may not be accepted by the terminal, independently of the padding. However, the detailed behavior in the end is implementation specific. For example, the current implementation is based on the JavaCard framework and the decryption procedure of PKCS#1 v1.5 merely throws an exception in case of incorrect padding and leaves it up to the higher-level program to treat this exception.

[11] The protocol explicitly notes that no error messages should be issued, but wrong implementations or side-channel attacks may reveal such information.

5.4 CBC-mode encryption

PLAID proposes to use CBC-mode encryption based on AES. The standard explicitly demands that the initialization vector IV is set to the all-zero string for both ^{e}STR2 (from the terminal to the card) and ^{e}STR3 (from the card to the terminal). This usage does not conform with standard practice, which demands the use of random IVs to achieve security against chosen plaintext attacks. As remarked before, the PLAID specification states that padding is only applied "if necessary" and is thus not compliant with ISO/IEC 9797-1 padding method 2, where padding is always applied. Indeed, this imprecision makes the standard unimplementable as currently specified, since there will be cases arising during decryption where it is not possible to discern whether padding should be removed or not. It is well known that CBC-mode encryption is especially vulnerable to padding oracle attacks [40] and that careful implementation is needed to avoid them. The lack of precision in this aspect of the PLAID specification does not bode well.

It is also now well understood in the cryptographic community that CBC-mode encryption does not offer sufficient integrity guarantees on its own to provide adequate security against active attacks. The usual solution is to add explicit integrity protection through the application of a MAC algorithm to the CBC-mode ciphertext. PLAID does not do so, and a justification for why this lack of integrity does not endanger security was requested in the comments of the previous ISO standard draft [21], but was not addressed in the latest version [19].

PLAID does offer mild forms of plaintext integrity. For example, STR2 contains the session key KeysHash computed as the hash of RND1 ‖ RND2, while STR3 contains DivData. These elements can be checked for after decryption by the relevant party, and this would detect some forms of adversarial plaintext manipulation through simple bit flipping in the corresponding ciphertexts ^{e}STR2 and ^{e}STR3. However, it is easy to see that an attacker can still manipulate other fields in STR2 and STR3 by bit flipping in ciphertexts (even with a fixed IV). While this lack of integrity has not led us to the discovery of specific attacks on PLAID, it is a worrying feature that could be easily avoided through the application of mainstream cryptographic design principles.

5.5 Entity authentication

Note that both parties, IFD (reader) and ICC (card), basically authenticate one another by proving knowledge of a

secret key. For the terminal, this is done via the secret RSA key, whereas the card uses its unique DivData and therefore unique key $\text{FAKey}^{(\text{Div})} = \text{AES}^{\text{FAKey}}_{\text{Encrypt}}(\text{DivData})$. The standard mechanism to do so would be either to compute a signature or a message authentication code for a random challenge, or to return, in clear, a nonce encrypted under the party's key.

PLAID follows the encryption-based approach. Yet the usual security argument for this type of authentication requires chosen-ciphertext security for the deployed encryption scheme. PLAID, on the other hand, uses two encryption schemes which are known not to provide this level of security, i.e., RSA-PKCS#1 v1.5 and plain AES-CBC encryption. This does not mean that the protocol is insecure and does not provide any form of entity authentication. However, one cannot infer security from known results but would instead need carefully constructed de novo arguments.

5.6 Payload insecurity

During the ISO standardization process, the PLAID protocol was changed to introduce an optional payload field in the third protocol message and the second message from the IFD to the card (see Step 5 in Fig. 1) [18]. The standard motivates the purpose of this payload field—this field should not be confused with the payload field in the last message by the card in Step 7 (Fig. 1)—for scenarios where, for example, a user enters a PIN and the verification should be done on the card. In this case, the PIN is to be sent to the card within the payload field [19, AnnexG]. The problem is that sensitive information send by the IFD can always be intercepted via a simple man-in-the-middle attack, assuming an adversary has corrupted an arbitrary card. Breaking a card allows the attacker to learn the diversified $\text{FAKey}^{(\text{Div})}$ of the card, the card's DivData field as well as the public part of one (or more) IAKeys. Thus, an adversary can simply replace the second message during an honest execution with one corresponding to the broken card. This will lead to the IFD encrypting the third message under the $\text{FAKey}^{(\text{Div})}$ of the broken card, and hence, if a PIN (or any other payload) is included, the adversary can trivially learn it by a simple decryption operation. Note that this problem is not due to the payload being sent in the third message, but that a user, when entering a PIN, cannot tell whether or not the terminal is actually communicating with his/her card. Therefore, PIN comparison on the card as proposed [19, AnnexG] is generically insecure due to the given attack scenario.

5.7 On the impossibility of key revocation

Although PLAID uses a public-key encryption system (i.e. RSA) during the initial authentication phase, the overall setup resembles more a symmetric setting where all static keys used by parties are exchanged during system setup (abstracting away the diversification procedure of PLAID). As a consequence, it is not possible to revoke any compromised keys within PLAID. In order to exemplify the resulting consequences, assume that an attacker is able to break into an IFD (terminal). The IFD contains a list of IAKeys and a list of FAKeys which thereby are revealed to the attacker. With this information, the attacker can generate arbitrary new cards with the capabilities of any of the KeySetIDs known by the broken IFD. Furthermore, there is no way to revoke the compromised keys in the system without issuing new cards, as the keys known by IFDs are hardcoded into the cards. Thus, even the break of a single IFD can lead to an entire PLAID setup becoming insecure.

5.8 Key legacy attack

A *key legacy attack* is related to the same issues allowing for the keyset fingerprinting attack, namely the lack of authentication in the list of keys used by a card and a terminal to establish a connection. Recall that the protocol specifies that the first commonly shared key in the list has to be used, even if there are other shared keys. This means that an adversary could force the card to use one particular key (among those supported by both card and terminal) by *reordering* the list of keys sent by the terminal in a man-in-the-middle fashion. This could be dangerous in case one or more of the keys in the system are compromised, or turn out to provide inferior security for any reason, even if the use of these keys is de-prioritized (e.g., by having the terminals set them always last in order of preference). We note that this type of attack was already mentioned in the national body comments to the first ISO draft [21], but remained unconsidered in the current version [19].

6 Responses of the ISO authority regarding technical aspects

We communicated our results to both the ISO 25185-1 project editor and to a contact person at the Department of Human Services. Aspects of the e-mail discussion that followed this communication can be found in a public, written response from the ISO project editor [15]. We stress, however, that we disagree with many points in this response, leading us to produce a statement concerning this response, which is also publicly available [10]. Here we report on technical aspects of the authority's response and leave discussion of other aspects to the next section, where we focus on our experience of the ISO standardization process.

The ISO project editor Graeme Freedman pointed out to us [16] (see also [15]) that card identifying information may also

be available to an adversary by other means, such as through the so-called Card Production Life Cycle (CPLC) data. The CPLC data contain information like serial numbers and manufacturers, uniquely identifying cards on a global scale. For privacy reasons, access to the CPLC data must therefore be restricted for PLAID. Indeed, the ISO draft standard itself already mentions this issue: "Consider switching off access to administrative applications from contactless interfaces, particularly ones which store unique card identification information such as the GlobalPlatform Card Production Life Cycle (CPLC) data." [19] Our results show, however, that even if one restricts access to such administrative data, then PLAID still leaks card and cardholder identifying information.

The editor's response in [15] concerning the ShillKey fingerprinting attack was to persist in insisting that the standard leaves open the implementation details about the ShillKey deployment. The report [15] states that "any change required to eliminate the attack (if desired) is solely up to the implementer/developer, since any implementation of ShillKey is interoperable with any other and the Standards are actually mute on how ShillKey is implemented and consequently how it is implemented is not an issue." We believe that a security-related standard should not introduce potential attack vectors by being ambiguous and leaving such important issues to developers. Otherwise, an implementation could be correct according to the ISO standard, but vulnerable to the ShillKey fingerprinting attack from our paper, allowing an attacker to identify cards.

We could not identify any comment in [15] concerning our keyset fingerprinting attack.

Most of our concerns presented in Sect. 5 were dismissed in [15] by stating that they are not supported by concrete attacks. For example, the lack of forward security of PLAID was countered in [15] by stating that "the Researchers have not described a method to obtain the keys in the first place," ignoring the fact that forward security exactly deals with the question of what security guarantees still hold *if* keys are leaked. However, the state of the art in cryptographic protocol design is now well beyond the approach of assuming that the absence of attacks is sufficient for judging a protocol to be sound. Instead, what is expected is rigorous formal analysis, using one or more of a variety of approaches (typically based on formal methods or the methodology of "provable security").

7 A cryptographer's perspective on the standardization process of PLAID

In this section, we consider our technical results in the context of PLAID's standardization in ISO. While some may argue that the main purpose of standardization is to provide interoperability and to increase productivity, ISO itself lists safety, reliability, and good quality as additional goals [20,37]. We view this as a clear indication that cryptographic protocols considered for ISO standardization should be also assessed according to their security guarantees.

Assuming one adopts the viewpoint that quality assurance should be an essential part of standardization, the question then arises of how this can be best accomplished. Below we review this question in light of the ISO standardization of PLAID, from our perspective as cryptographers who first became interested in the protocol out of scientific curiosity.

7.1 The pre-ISO phase

As already laid out in Sect. 1, the development of PLAID began in 2006 and was conducted by Centrelink, an agency of the Australian government's Department of Human Services (DHS). According to the report of PLAID's ISO project editor [15] on a preliminary version of this work, PLAID was in the following years subject to several private reviews, including by the Australian Defence Signals Directorate (now Australian Signals Directorate), the US National Institute of Standards and Technology (NIST), as well as by commercial vendors and workshops [hosted by the Australian DHS, NIST, Microsoft's security team, and the Asia Pacific Smart Card Association (APSCA)]. To the best of our knowledge, there are no publicly available results on the security of PLAID originating from these events.

Notwithstanding the above, we do think that a cryptographic protocol like PLAID which is supposed to become a national, or even an international standard, should receive thorough review by experts and that the results should be made public. Indeed, Centrelink's smart card architect in an interview in 2009 [35] agreed that "any cryptographic algorithm [...] which is supposed to be used for high security applications needs to be open and needs to be reviewed by the wider cryptographic community." Much to our surprise he continued by saying

> PLAID isn't a cryptographic algorithm, it's a protocol. PLAID [...] uses two cryptographic algorithms [RSA and AES]. [...] So, the actual cryptographic exchange [...] is based on two well established, well reviewed and considered secure algorithms [...].

While indeed PLAID is a protocol and not an algorithm, this does not obviate the need for a thorough (public) review. Indeed, our analysis here shows that equal care has to be taken when combining well-studied cryptographic algorithms (like RSA or AES) into a higher-level protocol. PLAID is not alone in this respect. For example, the long history of attacks on the most prominent cryptographic protocol to date, the Transport Layer Security (TLS) protocol [14], shows how delicate the

process of combining well-understood cryptographic primitives into a larger-scale protocol can be.

In 2010, PLAID was standardized as Australian standard AS 5185-2010 [38].

7.2 The ISO standardization process

In 2012, PLAID entered the ISO/IEC standardization process via the fast-track procedure as draft international standard (DIS) 25185-1 [18]. Inside ISO, working group WG 4 (Integrated circuit card with contacts) of subcommittee SC 17 (Cards and personal identification) within the Joint Technical Committee JTC 1 (Information technology) was entrusted with handling the standardization process. Given the necessary focus on the specification of a cryptographic protocol, we would suggest that cryptographic protocols like PLAID be assigned to cryptography-related working groups too, e.g., WG2 (Cryptography and security mechanisms) of SC27 (IT security techniques), in order to ensure a thorough examination of their cryptographic mechanisms.

The current ISO/IEC draft international standard version 25185-1.2 [19] was put forth in 2014 and incorporated minor changes to the original protocol for the purpose of alignment with other ISO standards. More important, in our opinion, were the changes and improvements requested in various formal comments made by several national bodies on the initial DIS that, as already mentioned in Sect. 5, were not resolved in version 25185-1.2.

For example, concerns that not authenticating the first message could lead to undetected manipulation of the KeySetIDs sent in that message were dismissed as an implementation issue. Our keyset fingerprinting attack (cf. Sect. 4) in contrast shows how to exploit this unauthenticated message to generically determine a card's capabilities, severely damaging the privacy properties of PLAID. In another case, CBC-mode encryption was wrongly claimed to provide message integrity, despite being otherwise noted and criticized in national body comments. Furthermore, remarks that there are no results indicating that RSA public keys cannot be recovered from RSA ciphertexts were written off with the irrelevant argument that, despite that concern, RSA is in use in most public-key infrastructures and also in TLS—ignoring that in the latter cases the public keys used are indeed made public and do not have to be kept secret for privacy reasons as is the case in PLAID. It turns out that these concerns were justified: RSA public keys can be accurately estimated from ciphertexts, and this directly allows our ShillKey fingerprinting attack (cf. Sect. 3) to trace cards and break their privacy. Finally, one editor comment challenged the usefulness of security proofs—demanded by some national bodies as a security guarantee—merely because RSA itself has no such security proof. We feel that such reasoning disregards the cryptographic community's substantial progress in establishing reliable frameworks for the analysis of security protocols over the last two decades.

Conducting this work as outsiders to the ISO standardization process, we received the impression that current procedures at ISO are not very amenable to encouraging public comments, e.g., from the academic community. Indeed, we took notice of the PLAID protocol and its status in the standards track by mere coincidence and had to first purchase the current DIS version in order to be able to begin investigating the protocol. After the completion of our analysis, we again struggled to find a formal mechanism to report our results back into the ISO process. Eventually, the kind facilitation of, in particular, German and UK national standardization body members enabled our findings to be taken into account in follow-up ISO discussions. Despite this cooperation, we still found it difficult to stay abreast of ongoing developments in PLAID's standardization, since the results of those discussions again remained inaccessible to the public.

We believe that a process that is more open toward public comment, especially by the academic community, would have the potential to result in a broader and more thorough examination of standards-track cryptographic algorithms and protocols. Prime examples of such processes are those followed by the Internet Engineering Task Force (IETF) in the development of widely deployed security protocols such as TLS and IPsec, and those adopted by NIST when developing the Advanced Encryption Standard (AES) and the new hash function standard SHA-3. Emphasizing this point, NIST recently announced its intention to open up its cryptography standardization processes even further and plans to handle public comments in a consistent, public, and accountable manner, seeking an even more extensive exchange with the academic community [27,33].

7.3 The aftermath of our work

At the time of writing, the ISO/IEC DIS [19] of PLAID is still in the "Enquiry stage" 40.60 (close of voting), as it was when we initially began our analysis. A preliminary version of this work, made public in September 2014 [12], has in the meantime been considered in discussions at several ISO JTC 1/SC 17/WG 4 meetings. We are, however, at this point in time unaware of any final decision having been taken on the ISO/IEC DIS of PLAID and, hence, its future as an ISO standard.

Besides gaining some attention at ISO and in the wider public arena, our work was in particular commented on by the Australian and ISO/IEC standard project editor for PLAID. The report produced by the project editor [15] claims to reveal errors in our work that render the described attacks both "mute" and "easily preventable." It also claims to identify mis-definitions and made-up privacy notions. We do not

wish to reiterate our view on that report here beyond the technical comments already made in Sect. 6, but instead refer the interested reader to our official response [10] in which we clarify why our concerns remain unchanged.

8 Conclusion

Our results show that PLAID has significant privacy weaknesses. The shill key attack and the keyset fingerprinting attack reveal card identifying information and, via access authorizations, information about the cardholder. As for entity authentication and the secrecy of established keys for subsequent communication, in several places the design of PLAID follows some uncommon strategies and reveals potential attack vectors, such as the lack of forward security. The case of PLAID also shows that standards should specify details thoroughly, in order to avoid vulnerable implementations. An example here is the ISO/IEC 9797-1 non-compliant CBC padding in PLAID, which potentially enables padding attacks (see our remark in Sect. 5).

We do not recommend the indiscriminate usage of PLAID in its current form, especially not for privacy-critical scenarios. While our proposed countermeasures seem to thwart our attacks on privacy, a more comprehensive analysis of the protocol in light of clearly stated security goals would be necessary. The PLAID description promises that the protocol should be scrutinized by "the most respected cryptographic organisations, as well as the broader cryptographic community" [7]. Unfortunately, we are not aware of any available documents in this regard. Indeed, standardization processes in general would benefit if supporting material, arguing the security of a proposal, were to be available at the time of evaluation.

As is, PLAID provides no privacy against active attacks, is not forward secure, and is ultimately based on symmetric-key cryptography (setting aside the use of public-key cryptography on top). One might expect that there should be easier approaches to obtaining secure authentication and key exchange protocols. Indeed, it seems that this problem and even approaches offering enhanced privacy have been discussed for a long time in the RFID community—see, for example, [25] for an early survey and [8] for a more recent one. Identifying specific protocol solutions from that area and discussing their security and efficiency features, however, is beyond the scope of our analysis of PLAID here.

Acknowledgments We thank Pooya Farshim for his contributions during the early stages of this paper, Andrew Waterhouse for providing insights on the ISO standardization process, and the anonymous reviewers for valuable comments. Marc Fischlin is supported by the Heisenberg grants Fi 940/3-1 and Fi 940/3-2 of the German Research Foundation (DFG). Tommaso Gagliardoni and Felix Günther are supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE. Felix Günther and Giorgia Azzurra Marson are supported by the DFG as part of the CRC 1119 CROSSING. Giorgia Azzurra Marson and Arno Mittelbach are supported by the Hessian LOEWE excellence initiative within CASED. Kenneth G. Paterson and Jean Paul Degabriele are supported by the Engineering and Physical Sciences Research Council (EPSRC) Leadership Fellowship EP/H005455/1.

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO 1993, pp. 232–249. Springer Berlin, Hidelberg (1994)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Eurocrypt 2000, pp. 139–155. Springer Berlin, Hidelberg (2000)
3. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: ASIACRYPT 2001, pp. 566–582. Springer Berlin, Hidelberg (2001)
4. Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.Y., Zanella Béguelin, S.: Proving the TLS handshake secure (as it is). 235–255 (2014). doi:10.1007/978-3-662-44381-1_14
5. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. 1–12 (1998)
6. Brzuska, C., Fischlin, M., Smart, N.P., Warinschi, B., Williams, S.C.: Less is more: relaxed yet composable security notions for key exchange. Int. J. Inf. Secur. **12**(4), 267–297 (2013)
7. Centrelink: Protocol for Lightweight Authentication of Identity (PLAID)—Logical Smartcard Implementation Specification PLAID Version 8.0—Final. http://www.humanservices.gov.au/corporate/publications-and-resources/plaid/technical-specification (2009)
8. Coisel, I., Martin, T.: Untangling RFID privacy models. J. Comput. Netw. Commun. doi:10.1155/2013/710275
9. Dagdelen, Ö., Fischlin, M., Gagliardoni, T., Marson, G.A., Mittelbach, A., Onete, C.: A cryptographic analysis of OPACITY—(extended abstract). pp. 345–362 (2013). doi:10.1007/978-3-642-40203-6_20
10. Degabriele, J.P., Fehr, V., Fischlin, M., Gagliardoni, T., Günther, F., Marson, G.A., Mittelbach, A., Paterson, K.G.: Response to "Nit-Picking PLAID AS & ISO Project Editors Report into 'Unpicking Plaid' ". Cryptology ePrint Archive Forum, http://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/pdf/plaid-editorreport-response.pdf (2014)
11. Degabriele, J.P., Fehr, V., Fischlin, M., Gagliardoni, T., Günther, F., Marson, G.A., Mittelbach, A., Paterson, K.G.: Unpicking PLAID—a cryptographic analysis of an ISO-standards-track authentication protocol. In: 1st International Conference on Research in Security Standardisation (SSR 2014). Springer, Lecture Notes in Computer Science, vol. 8893, pp. 1–25 (2014)
12. Degabriele, J.P., Fehr, V., Fischlin, M., Gagliardoni, T., Günther, F., Marson, G.A., Mittelbach, A., Paterson, K.G.: Unpicking PLAID—a cryptographic analysis of an ISO-standards-track authentication protocol. Cryptology ePrint Archive, Report 2014/728. http://eprint.iacr.org/ (2014)
13. Department of Human Services: Protocol for Lightweight Authentication of Identity (PLAID). (2014). http://www.humanservices.gov.au/corporate/publications-and-resources/plaid/
14. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard). http://www.ietf.org/rfc/rfc5246.txt, updated by RFCs 5746, 5878, 6176 (2008)
15. Freedman, G.: Nit-Picking PLAID: AS & ISO Project Editors Report into "Unpicking Plaid". Cryptology ePrint Archive Forum. https://dl.dropboxusercontent.com/u/41736374/UnpickingReport%20V1.pdf (2014)

16. Freedman, G.: Personal communication by e-mail (2014)
17. Giesen, F., Kohlar, F., Stebila, D.: On the security of TLS renegotiation. In: ACM Conference on Computer and Communications Security, pp. 387–398. ACM, New York (2013)
18. ISO: DRAFT INTERNATIONAL STANDARD ISO/IEC DIS 25185–1 Identification cards—Integrated circuit card authentication protocols—Part 1: Protocol for Lightweight Authentication of Identity. International Organization for Standardization, Geneva (2012)
19. ISO: DRAFT INTERNATIONAL STANDARD ISO/IEC DIS 25185-1.2 Identification cards—Integrated circuit card authentication protocols—Part 1: Protocol for Lightweight Authentication of Identity. International Organization for Standardization, Geneva (2014)
20. ISO: Benefits of international standards. (2015). http://www.iso.org/iso/home/standards/benefitsofstandards.htm
21. ISO 25185–1 Editor (2013) Disposition of comments on ISO/IEC 25185–1 Protocol for a lightweight authentication of devices
22. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. 273–293 (2012)
23. Jager, T., Schinzel, S., Somorovsky, J.: Bleichenbacher's attack strikes again: breaking PKCS#1 v1.5 in XML encryption. 752–769 (2012)
24. Johnson, R.: Estimating the size of a population. Teach. Stat. **16**(2), 50–52 (1994). http://www.mcs.sdsmt.edu/rwjohnso/html/tank.pdf
25. Juels, A.: RFID security and privacy: a research survey. IEEE J. Selected Areas Commun. **24**(2), 381–394 (2006)
26. Kaliski, B.: PKCS#1: RSA Encryption Version 1.5. RFC 2313 (1998)
27. Kelsey, J.: Dual EC DRBG and NIST crypto process review. In: Invited talk at the Real World Cryptography Workshop 2015, January 7–9, London (2015)
28. Kiat, K.H., Run, L.Y.: An analysis of OPACITY and PLAID protocols for contactless smart cards. Master's thesis, Naval Postgraduate School, Monterey (2012)
29. Kline, R.: Improving contactless security is goal of emerging PLAID project. http://secureidnews.com/news-item/improving-contactless-security-is-goal-of-emerging-plaid-project/, secureIDNews (2010)
30. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: a systematic analysis. (2013). doi:10.1007/978-3-642-40041-4_24
31. Meyer, C., Somorovsky, J., Weiss, E., Schwenk, J.: Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks. In: 23rd USENIX Security Symposium (USENIX Security 14), USENIX Association, San Diego (2014). https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/meyer
32. National Institute of Standards and Technology: Protocol for Lightweight Authentication of Identity (PLAID) Workshop (2009). http://csrc.nist.gov/news_events/plaid-workshop/
33. National Institute of Standards and Technology: Cryptographic Standards and Guidelines Development Process (Second Draft). National Institute of Standards and Technology Interagency Report 7977. http://csrc.nist.gov/publications/drafts/nistir-7977/nistir_7977_second_draft.pdf (2015)
34. Rifà-Pous, H., Herrera-Joancomartí, J.: Computational and energy costs of cryptographic algorithms on handheld devices. Future Internet **3**(1), 31–48 (2011)
35. Riskybiz: Risky Business 106—Centrelink's new PLAID auth protocol. http://risky.biz/netcasts/risky-business/risky-business-106-centrelinks-new-plaid-auth-protocol (2009)
36. Sakurada, H.: Security evaluation of the PLAID protocol using the ProVerif tool. http://crypto-protocol.nict.go.jp/data/eng/ISOIEC_Protocols/25185-1/25185-1_ProVerif.pdf (2013)
37. Sanders, T.: The Aims and Principles of Standardization. International Organization for Standardization—ISO (1972)
38. Standards Australia: AS 5185-2010 Protocol for Lightweight Authentication of IDentity (PLAID). Standards Australia (2010)
39. Taylor, J.: Centrelink ID protocol still in trial phase. http://www.zdnet.com/centrelink-id-protocol-still-in-trial-phase-1339336953/, zDNet (2012)
40. Vaudenay, S.: Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS. pp. 534–546 (2002)
41. Watanabe, D.: Security analysis of PLAID. http://crypto-protocol.nict.go.jp/data/eng/ISOIEC_Protocols/25185-1/25185-1_Scyther.pdf (2013)