A Cryptographic Analysis of OPACITY
(Extended Abstract)

Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni,
Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete

Darmstadt University of Technology, Germany
`www.cryptoplexity.de`

Abstract. We take a closer look at the Open Protocol for Access Control, Identification, and Ticketing with privacY (OPACITY). This Diffie–Hellman-based protocol is supposed to provide a secure and privacy-friendly key establishment for contactless environments. It is promoted by the US Department of Defense and meanwhile available in several standards such as ISO/IEC 24727-6 and ANSI 504-1. To the best of our knowledge, so far no detailed cryptographic analysis has been publicly available. Thus, we investigate in how far the common security properties for authenticated key exchange and impersonation resistance, as well as privacy-related properties like untraceability and deniability, are met.

OPACITY is not a single protocol but, in fact, a suite consisting of two protocols, one called Zero-Key Management (ZKM) and the other one named Fully Secrecy (FS). Our results indicate that the ZKM version does not achieve even very basic security guarantees. The FS protocol, on the other hand, provides a decent level of security for key establishment. Yet, our results show that the persistent-binding steps, for re-establishing previous connections, conflict with fundamental privacy properties.

1 Introduction

OPACITY is short for the Open Protocol for Access Control, Identification, and Ticketing with privacY. It is basically a Diffie–Hellman-based protocol to establish secure channels in contactless environments. According to Eric Le Saint of the company ActivIdentity, co-inventor in the patent application [47], the development has been sponsored by the US Department of Defense [48]. The inventors have declared the contributions to OPACITY to be a statutory invention with the United States Patent and Trademark Office, essentially allowing royalty-free and public usage of the contribution. The protocol has been registered as an ISO/IEC 24727-6 authentication protocol [27] and is specified in the draft ANSI 504-1 national standard (GICS) [24]. Informal yet outdated descriptions are available through the homepage of the Smart Card Alliance [3].[1]

[1] We stress that none of the authors of the present paper has been involved in the development of OPACITY, or is employed by ActivIdentity, or is supported by a non-academic governmental agency for conducting this research.

1.1 Security Assessment of OPACITY

As Eric Le Saint emphasizes in his description of OPACITY [48], "This protocol was designed expressly to remove the usage restrictions on contactless transactions while still delivering high performance security and privacy." Surprisingly, we are not aware of any profound and public cryptographic analysis of the protocol, including clear claims about security and privacy goals. The best effort, in terms of the Smart Card Alliance, seems to be compliance with standards [3]:

> "The protocol strictly follows U.S. government and international standards. It has been assessed for compliance with the NIST standard for key establishment protocols (SP 800-56A). As a consequence, further protocol design reviews are unnecessary prior to FIPS 140-2 security certification."

It is of course not the case —and we do not think that the Smart Card Alliance statement suggests so— that compliance with SP 800-56A, or certification according to FIPS 140-2, instantaneously gives strong cryptographic security guarantees. The NIST document SP 800-56A [41] only provides useful but, nonetheless, high-level *recommendations* for key-establishment schemes based on the discrete logarithm problem, and specifies some schemes from ANSI X9. To the best of our knowledge, it has not been shown formally yet under which conditions protocols complying with SP 800-56A are also cryptographically secure (in whatever sense). This is particularly true as OPACITY supports renegotiation techniques and also states privacy enhancement as an additional goal. Neither property is discussed in SP 800-56A.

Similarly, even if OPACITY was FIPS 140-2 certified and thus checked by an accredited authority, this does not necessarily imply strong security guarantees either. An obvious testimony to this argument are the easy attacks on FIPS 140-2 level 2 certified USB memory tokens where access was always granted for a fixed string, independently of the password [17,18]. Certification according to FIPS 140-2, and this is acknowledged in the standard, only intends "to maintain the security provided by a cryptographic module" in the utilized environment; the "operator of a cryptographic module is responsible for ensuring that the security provided by the module is sufficient." (see [39]).

Hence, we believe that OPACITY deserves a closer cryptographic look. Clearly, there are many practical protocols which lack such an analysis, or have at least not been scrutinized publicly. What makes OPACITY a worthwhile object for a cryptographic analysis is:

– OPACITY is standardized and may thus be deployed extensively in the near future. This is all the more true as it is a general purpose protocol, suitable, for instance, for use in access control for buildings, but also for ticketing in transport systems [48].
– OPACITY does not seem to be deployed broadly yet. It is our firm belief that protocols should be rigorously analyzed *before* they are actually utilized, in order to prevent damage caused by weaknesses discovered after deployment. Furthermore, patching a popular protocol in use is often intricate and progresses slowly (see the example of MD5-based certificates [51]).

– OPACITY still has a decent level of abstract description complexity. While nonetheless being quite complex underneath, especially taking into account different execution modes such as renegotiation steps (called persistent binding for OPACITY), this should be contrasted with similar protocols like SSL/TLS where conducting cryptographic proofs is tedious; such works often focus on particular parts or (modified) versions of the protocol [22,38,45,29].

Another point, which we initially thought speaks for OPACITY, is the availability of an open source implementation on Source Forge [43]. Unfortunately, as later confirmed by the developers of OPACITY [49], this implementation seems to refer to an outdated version. The differences were sufficiently large to determine us not to investigate the source code on how the cryptographic concepts are realized; nonetheless, we occasionally consulted the source code in order to extrapolate, in case some specification details were missing.

1.2 Our Results

OPACITY is a family of Diffie-Hellman key-exchange protocols based on Elliptic Curve Cryptography. It comes in two versions, called Zero-Key Management (O-ZKM) and Full Secrecy (O-FS). The first name is due to the fact that the terminal does not need to maintain registered public keys. As such, the parties in the O-ZKM protocol run a Diffie–Hellman based key-exchange protocol using an ephemeral key on the terminal's side and a static (presumably on-card generated) key for the card. The experienced reader may immediately spot the weakness in this approach: since the terminal only uses ephemeral keys, anyone can in principle impersonate the terminal and successfully initiate a communication with the card. Jumping ahead, we note that we can neither achieve a weaker notion of one-way authenticated key exchange [23] with this protocol. Before we go into further details of the security of the protocols, let us point out that the second protocol, O-FS, uses long-term keys on both sides and runs two nested Diffie–Hellman protocols, each one with the static key of the parties and an ephemeral key from the other party. This at least rules out obvious impersonation attacks.

Targeted security properties. Obviously, OPACITY aims at establishing a secure channel between the parties and to provide some form of entity authentication, especially impersonation resistance against malicious cards. Yet, at the same time, OPACITY also seems to target privacy properties. There seems to be a general and rough agreement what we expect from a "secure" key-exchange protocol, despite technical differences in the actual models [5,14]. We opted for the common Bellare-Rogaway (BR) model for key exchange but we also consider key-compromise impersonation resistance and leakage of ephemeral secrets in the related eCK model [34] in the full version [16].[2] We note that cryptographic analysis of similar key exchange protocols, such as for NIST's KEA [4,35,32] or

[2] Let us mention here that the protocols cannot be proven secure in the eCK model.

for the ANSI X9.63 specified UM protocols [4,37] cannot be transferred to OPACITY, as these protocols differ in security-relevant details and do not support renegotiation (and do not touch privacy issues); we comment on the differences in Section 3.3.

The privacy requirements for OPACITY are, however, less clear than the ones for key secrecy. This is all the more true as they are never specified in the accompanying documents. An earlier version of the OPACITY protocol description [50] mentions the following two goals for the O-FS protocol:

– "The OPACITY protocol does not divulge any data that allows the correlation of two protocol executions with same ICC [card] during an OPACITY session."
– "The OPACITY protocol does not divulge any identifier associated to a particular ICC or card holder during an OPACITY session."

The first requirement resembles the well-known notion of *untraceability* for protocols. We thus adopt the framework of Ouafi and Phan [44] which can be seen as a "BR-like" definition of the Juels and Weiss model [30] matching our approach for the key-agreement part. We do not explore stronger (simulation-based) models like the one in [36] as the protocols fail to provide security even in these more basic models.

The second desirable privacy property seems to be weaker in that it allows linkability in principle, but tries to hide the card's or the card holder's identity. We therefore introduce a notion called *identity hiding* which also follows the BR attack model, but instead of preventing the adversary from distinguishing between two cards —as for untraceability— we only guarantee that one cannot deduce the card's certificate (i.e., its identity). Note that, some authors such as [23], use the term identity hiding to denote the fact that the peer does not learn the partner's identity before the execution ends; our notion here coincides with this idea for the OPACITY protocols.

Basically, identity hiding as defined here is similar to recognizing a person without knowing the person's name. By contrast, untraceability is similar to not being able to tell that a particular person has been seen twice (this is independent of a person's name). Clearly, identity hiding gives weaker anonymity guarantees than untraceability or anonymity of credential systems [9,15]. Even direct anonymous attestation [10] or cross-domain anonymity as in the case of the German identity card [6] support linkability only within specified domains but are otherwise untraceable. Hence, the notion of identity hiding should be taken with caution.

Another desirable privacy property for OPACITY may be *deniability* [21], that is, the inability to use transcripts of communications as proofs towards third parties. Although not explicitly listed as a goal, it may be advantageous for a multi-purpose card protocol like OPACITY. There are different approaches and levels of deniability [8,20,19,23]; in light of what OPACITY can achieve we focus on a very basic level protecting only against abuse of transcripts between honest parties (denoted *outsider deniability* here).

Table 1. Security properties of the OPACITY protocol

	OPACITY-ZKM	**OPACITY-FS**
BR key secrecy	(only passive and if modified)	✓
+ forward secrecy	—	(only weak)
Impersonation Resistance	(only cards)	(only cards)
Untraceability	—	(only w/o persistent binding)
Identity Hiding	—	✓
(Outsider) Deniability	(only w/o persistent binding)	(only w/o persistent binding)

Finally, the goal of the OPACITY protocols is to establish a key which is subsequently used to secure communication between the card and the terminal. As such, one is of course interested in the security of the secure messaging protocol of OPACITY as well as in the overall composition of the key-agreement protocol and the secure messaging. Here, we rely on recent results for the secure composition of BR-secure key-exchange protocols [12,11]. We next discuss and illustrate exactly which security levels are achieved by OPACITY.

Achieved security properties. Our results are summarized in Table 1. The protocol O-ZKM cannot achieve BR-security against malicious terminals. Even for passive adversaries (merely observing executions between honest parties) the protocol is not secure; it *does* fulfill BR-security *only* after a slight modification of the protocol. The O-FS protocol achieves BR-security under the Gap Diffie–Hellman assumption [42] in the random-oracle model, assuming that the underlying cryptographic primitives are secure.[3] As for impersonation resistance, since the terminal does not authenticate towards the card, we can only hope to achieve security against malicious cards. This is met for both protocols given that the underlying message authentication scheme is secure.

As far as privacy is concerned, we show that neither protocol achieves untraceability nor even a weakened form of untracebility. For O-ZKM this is quite clear, as parts of the card's certificate are sent in clear. For O-FS the certificate is encrypted, yet we show that it is easy to desynchronize the cards' states and hence, due to persistent binding, to mount privacy attacks via desynchronization attacks. If, on the other hand, we only consider O-FS without renegotiation (and thus without any accumulated state), untraceability is met. Note that this is not the case for O-ZKM, that is, even without persistent binding (i.e., renegotiation) O-ZKM is traceable. For identity hiding, we can show that it is met by O-FS but not by O-ZKM.[4]

Concerning (outsider) deniability, we again only give a conditional result: OPACITY without persistent binding can be proved (outsider) deniable both

[3] This apparently innocent assumption about the security of the primitives has a hidden layer underneath. OPACITY is not fully specified in the standards and operates in some arguably doubtful modes, so this assumption must be taken with caution. We comment on this later.

[4] Note that O-ZKM contains steps which indicate that some form of identity hiding was aimed for: parts of the identity are only sent encrypted. Nevertheless an easy attack exists which we present in the full version [16].

for O-FS and O-ZKM. Persistent binding does, however, allow for simple attacks in many of the existing models for deniability, as well as, in our rather weak model of outsider deniability. Furthermore, persistent binding opens the door to physical attacks, for example, by simply comparing the state of the physical registers containing the persistent binding information of a terminal and card, one could extract high-confidence proofs that the card and terminal have been partnered in at least a single session.

An extended abstract. This version is an extended abstract of our results. Due to space restrictions we had to sacrifice some details. The interested reader is kindly referred to the full version of this work [16] for more details.

2 Security Model

2.1 Key Secrecy (Authenticated Key Exchange)

We analyze OPACITY with respect to key secrecy in the real-or-random security model by Bellare and Rogaway [5]. Roughly speaking, an adversary should not be able to tell apart a genuine session key from a key uniformly sampled from the key space. The security model defines so-called sessions, describes an attack model, and shows a winning condition for an adversary.

Our model (described in detail in the full version) follows the one in [5] closely. The adversary controls the network and can interact with the parties —(instances of) terminals or cards— through Execute and Send queries: the former is used to run the protocol between an honest terminal and an honest card, and the latter enables the adversary to send protocol messages to honest parties. We assume that the adversary can choose whether an honest terminal should request to reconnect via persistent binding or not (in the protocol this can be done quite easily by an active adversary which can alter or add appropriate bits to the terminal's first message —see the protocol description in Section 3). As usual, the adversary can test sessions (via a Test oracle), ask to reveal session keys, and, for forward-secure versions, corrupt parties (thus receiving the party's long-term secret key and state, i.e., the information stored for persistent binding and, for strong corruption, also the party's random coins and ephemeral secrets). For key secrecy it suffices to consider a single Test-query [1], in which the adversary receives either the true session key or a random key, depending on a random bit b. We also assume that the adversary can register any chosen public key on behalf of corrupted parties, possibly even keys already registered by honest parties, and receive certificates for such keys from the (trusted) certification authority. We assume that identities in certificates are unique.

We specify intended partners by partner id's pid and sessions by session id's sid (defined according to the protocol description). Two sessions are *partnered* if they have both accepted and output the same session id. We assume that untampered executions between honest parties generate the same sid and the same session key. The adversary can only test sessions in which she cannot trivially deduce the session key. A crucial notion for this security definition is that of *freshness*. Informally, an instance P_i is *fresh* (with respect to authenticated key-exchange —AKE— security), iff: (i) the adversary has not asked to reveal the key of that instance, nor of a partnered instance; (ii) the adversary has made no corruption queries, and (iii) neither P_i nor the intended partner pid output by P_i is adversarially controlled (in particular, their secret keys were not registered by the adversary). For the study of forward secrecy we need to adapt the notion of freshness to allow corruptions under certain restrictions. We refer to the full version for an introduction to forward secrecy and an analysis of OPACITY with respect to it.

Eventually, the adversary $\mathcal{A}$ outputs a guess b' for the secret bit b used in the Test-oracle. The adversary is *successful* iff: $b = b'$, and the instance P_i in the Test-oracle is fresh. We are interested in the advantage of the adversary over the simple guessing probability of 1/2. We usually consider security relative to the adversary's parameters, such as its running time t, the number q_e of initiated executions of protocol instances of Π, and, modeling the key derivation function as a random oracle, the number q_h of random oracle queries of the adversary. For some of the security notions we also make the number of Test queries explicit through a parameter q_t.

Definition 1 (Key Secrecy). *We call a protocol Π, running for security parameter λ, (t, q_e, q_h, ϵ)-secure if no algorithm running in time t, invoking q_e instances of Π and making at most q_h queries to the random oracle can win the above experiment with probability greater than $\frac{1}{2} + \epsilon$. We call the value $\left|\Pr[\mathcal{A}\ \mathsf{wins}] - \frac{1}{2}\right|$ the* advantage *of the algorithm $\mathcal{A}$, and we denote the maximum over all (t, q_e, q_h)-bounded $\mathcal{A}$ by $\mathbf{Adv}_{\Pi}^{ake}(t, q_e, q_h)$.*

The BR model is a strong security model providing confidentiality of agreed session keys and their authenticity (i.e., at most one partner shares the derived keys). Furthermore, one can also show forward secrecy by adjusting the freshness notion. However, as LaMacchia et al. [34] pointed out, certain attacks, such as key-compromise impersonation and leakage of ephemeral secrets, are not covered by the BR model. We discuss these properties and analyze the OPACITY protocols with respect to them in the full version.

2.2 Impersonation Resistance

The notion of authenticated key exchange ensures that only the intended partner can compute the session key (i.e. an adversary that is not partnered with a specific partner in some session, cannot compute that session's key). For some application scenarios, however, we may also need that the terminal can be sure of the card's identity. This could be guaranteed by subsequent use of the computed session keys, but this is application-dependent. Impersonation resistance, as defined here, gives instead direct guarantees and is closer to the security of identification schemes. We give a strong definition based on the BR framework, which includes common properties like passive and active security for identification schemes. Still, note that we only consider impersonation by malicious cards to a terminal (and not that of malicious terminals to a card).

The attack model for impersonation resistance resembles AKE, but this time there are no Test-queries. The adversary's goal is to impersonate an honest card, without using trivial Man-in-the-Middle relaying attacks or making the terminal accept a card which has not been issued (resp. certified) by the certification authority $\mathcal{CA}$. More formally, the terminal must accept in some session sid for partner id pid, such that (a) pid is not adversarially controlled, and (b) there is no accepting card session for honest card pid with the same sid (including also the case that party pid has not been registered with a public key). If this happens we say that the adversary wins.

Definition 2 (Impersonation Resistance). *We call a protocol Π, running for security parameter λ, (t, q_e, q_h, ϵ)-impersonation resistant if no algorithm running in time t, invoking q_e instances of Π and making at most q_h queries to the random oracle can win the above experiment with probability greater than ϵ. We call the value* $\Pr[\mathcal{A}\ \mathsf{wins}]$ *the advantage of the algorithm $\mathcal{A}$, and we denote the maximum over all (t, q_e, q_h)-bounded $\mathcal{A}$ by $\mathbf{Adv}_{\Pi}^{ir}(t, q_e, q_h)$.*

2.3 Privacy for Key Exchange

Privacy in cryptography comes in many different flavors. The OPACITY documentation does not clarify exactly which properties the protocol is aiming for. We consider two reasonable notions, untraceability and identity hiding, and discuss the latter below. Due to space restrictions, the analysis of OPACITY in terms of untraceability is deferred to the full version, where we also define deniability for KE and show that, for OPACITY, it does imply untraceability in the restricted case where the renegotiation mode is not used.

Identity Hiding. Intuitively, an adversary against untraceability should not be able to link two sessions run by the same card. A weaker notion, called *identity hiding*, only stipulates that an adversary is unable to know *which* card authenticates (though she may know that she has seen this card authenticate before). Thus, untraceability hides both the identity (i.e., the certificate) of the card and its history (e.g., its state). By contrast, identity hiding only hides the certificate.

We use the identical security model as for key exchange, but with one exception: we assume a special card $\mathcal{C}^*$ exists, for which two certified key-pairs $(\mathsf{sk}_0^*, \mathsf{pk}_0^*, \mathrm{cert}_0^*)$, $(\mathsf{sk}_1^*, \mathsf{pk}_1^*, \mathrm{cert}_1^*)$ are generated under (potentially different) identities. The adversary is initially given the certificates and public keys of all honest parties except for $\mathcal{C}^*$, together with the assignment of the keys and certificates to the cards. The adversary also receives the two pairs $(\mathsf{pk}_0^*, \mathrm{cert}_0^*)$, $(\mathsf{pk}_1^*, \mathrm{cert}_1^*)$. At the start of the game, a bit b is flipped and $\mathcal{C}^*$ is instantiated with $(\mathsf{sk}_b^*, \mathsf{pk}_b^*, \mathrm{cert}_b^*)$. When the Test oracle is queried, it returns the handle for card $\mathcal{C}^*$, allowing the adversary to access this card by using all the previous oracles, apart from $\mathsf{Corrupt}$. The adversary must predict the bit b, i.e. it must learn whether card $\mathcal{C}^*$ is associated with the left or right key pair. The only restriction is that the partner id pid output in any of the Test sessions is always an identity of an honest terminal (if the terminal is malicious the adversary trivially decrypts the encrypted certificate). Furthermore, no $\mathsf{Corrupt}$ queries must be issued to terminals.

Note that in this model the adversary cannot choose the target key pairs adaptively (having received a list of valid certificates). However, our approach is equivalent (up to a factor equal to the square of the number of the certificates) with a model using adaptive selection.

Definition 3 (Identity Hiding). *We call a protocol Π, running for security parameter λ, $(t, q_e, q_t, q_h, \epsilon)$-*identity-hiding *if no algorithm $\mathcal{A}$ running in time t, invoking q_e instances of Π, including q_t* Test*-sessions, and making at most q_h queries to the random oracle, can win the above experiment with probability greater than $\frac{1}{2} + \epsilon$. We call the value $\left|\Pr[\mathcal{A}\ \mathsf{wins}] - \frac{1}{2}\right|$ the* advantage *of the algorithm $\mathcal{A}$, and we denote the maximum over all (t, q_e, q_t, q_h)-bounded $\mathcal{A}$ by $\mathbf{Adv}_{\Pi}^{\textit{id-hide}}(t, q_e, q_t, q_h)$.*

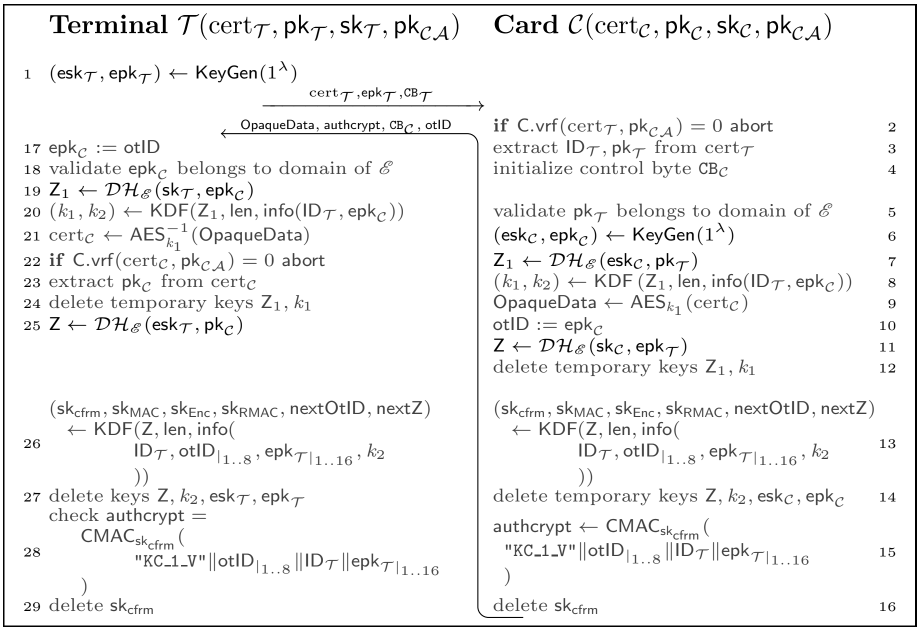
3 The OPACITY Protocols

The OPACITY suite contains two key-exchange protocols, one called *OPACITY with Zero-Key Management* (O-ZKM), the other *OPACITY with Full Secrecy* (O-FS). Both protocols allow a terminal $\mathcal{T}$ and a card $\mathcal{C}$ to agree upon session keys $\mathsf{sk}_{\mathsf{MAC}}, \mathsf{sk}_{\mathsf{Enc}}, \mathsf{sk}_{\mathsf{RMAC}}$ (for command authentication, encryption, and response authentication). Note, however, that though subsumed under the same protocol suite, the two protocols are nonetheless quite different, the main difference being that O-ZKM has only one-sided authentication, i.e., the card authenticates to the terminal but not vice versa. Due to space restrictions, in this extended abstract we only present a slightly simplified version of O-FS (see Figure 1); both O-ZKM and the complete O-FS are discussed in detail in the full version. The theorems presented in this extended abstract apply, however, to the full O-FS-protocol. We discuss related protocols in Section 3.3.

3.1 Protocol Descriptions

Both protocols (O-ZKM and -FS) consist of two rounds, the first one initialized by the terminal. Our description closely follows the original formulation in the standards. We make, however, minor changes in notation so as to simplify the diagram and improve legibility. We also change some variable names to be more compliant to standard cryptographic descriptions of protocols. We give a shortened description of the O-FS protocol, without renegotiation, in Figure 1.

From a bird's-eye view the O-FS protocol works as follows. Both the terminal and the card hold a certified key pair $(\mathsf{pk}_{\mathcal{T}}, \mathsf{sk}_{\mathcal{T}})$ and $(\mathsf{pk}_{\mathcal{C}}, \mathsf{sk}_{\mathcal{C}})$, respectively. The protocol works over a suitable elliptic curve $\mathscr{E}$; as such, secret keys are the discrete logarithms of the corresponding public keys (for some generator G). Both parties also generate an ephemeral key pair for each session, denoted by $(\mathsf{epk}_{\mathcal{T}}, \mathsf{esk}_{\mathcal{T}})$ and $(\mathsf{epk}_{\mathcal{C}}, \mathsf{esk}_{\mathcal{C}})$. The terminal first transmits its public keys $\mathsf{pk}_{\mathcal{T}}$ (encapsulated in the certificate) and $\mathsf{epk}_{\mathcal{T}}$, together with a control byte $\mathtt{CB}_{\mathcal{T}}$ for specifying different modes and for indicating a renegotiation request. The first Diffie-Hellman key is computed via the static key $\mathsf{pk}_{\mathcal{T}}$ of the terminal and the



```
Terminal $\mathcal{T}(\mathrm{cert}_\mathcal{T}, \mathsf{pk}_\mathcal{T}, \mathsf{sk}_\mathcal{T}, \mathsf{pk}_{\mathcal{CA}})$        Card $\mathcal{C}(\mathrm{cert}_\mathcal{C}, \mathsf{pk}_\mathcal{C}, \mathsf{sk}_\mathcal{C}, \mathsf{pk}_{\mathcal{CA}})$

1  $(\mathsf{esk}_\mathcal{T}, \mathsf{epk}_\mathcal{T}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
        $\xrightarrow{\mathrm{cert}_\mathcal{T}, \mathsf{epk}_\mathcal{T}, \mathtt{CB}_\mathcal{T}}$
                                          if $\mathsf{C.vrf}(\mathrm{cert}_\mathcal{T}, \mathsf{pk}_{\mathcal{CA}}) = 0$ abort   2
                                          extract $\mathsf{ID}_\mathcal{T}, \mathsf{pk}_\mathcal{T}$ from $\mathrm{cert}_\mathcal{T}$   3
                                          initialize control byte $\mathtt{CB}_\mathcal{C}$   4
                                          validate $\mathsf{pk}_\mathcal{T}$ belongs to domain of $\mathscr{E}$   5
                                          $(\mathsf{esk}_\mathcal{C}, \mathsf{epk}_\mathcal{C}) \leftarrow \mathsf{KeyGen}(1^\lambda)$   6
                                          $\mathsf{Z}_1 \leftarrow \mathcal{DH}_\mathscr{E}(\mathsf{esk}_\mathcal{C}, \mathsf{pk}_\mathcal{T})$   7
                                          $(k_1, k_2) \leftarrow \mathsf{KDF}(\mathsf{Z}_1, \mathsf{len}, \mathsf{info}(\mathsf{ID}_\mathcal{T}, \mathsf{epk}_\mathcal{C}))$   8
                                          $\mathsf{OpaqueData} \leftarrow \mathsf{AES}_{k_1}(\mathrm{cert}_\mathcal{C})$   9
                                          $\mathsf{otID} := \mathsf{epk}_\mathcal{C}$   10
                                          $\mathsf{Z} \leftarrow \mathcal{DH}_\mathscr{E}(\mathsf{sk}_\mathcal{C}, \mathsf{epk}_\mathcal{T})$   11
                                          delete temporary keys $\mathsf{Z}_1, k_1$   12
                                          $(\mathsf{sk_{cfrm}}, \mathsf{sk_{MAC}}, \mathsf{sk_{Enc}}, \mathsf{sk_{RMAC}}, \mathsf{nextOtID}, \mathsf{nextZ})$
                                            $\leftarrow \mathsf{KDF}(\mathsf{Z}, \mathsf{len}, \mathsf{info}($
                                                $\mathsf{ID}_\mathcal{T}, \mathsf{otID}_{|1..8}, \mathsf{epk}_{\mathcal{T}|1..16}, k_2$   13
                                                $))$
                                          delete temporary keys $\mathsf{Z}, k_2, \mathsf{esk}_\mathcal{C}, \mathsf{epk}_\mathcal{C}$   14
                                          $\mathsf{authcrypt} \leftarrow \mathsf{CMAC}_{\mathsf{sk_{cfrm}}}($
                                            "KC_1_V"$\|\mathsf{otID}_{|1..8}\|\mathsf{ID}_\mathcal{T}\|\mathsf{epk}_{\mathcal{T}|1..16}$   15
                                            $)$
                                          delete $\mathsf{sk_{cfrm}}$   16
        $\xleftarrow{\mathsf{OpaqueData}, \mathsf{authcrypt}, \mathtt{CB}_\mathcal{C}, \mathsf{otID}}$
17 $\mathsf{epk}_\mathcal{C} := \mathsf{otID}$
18 validate $\mathsf{epk}_\mathcal{C}$ belongs to domain of $\mathscr{E}$
19 $\mathsf{Z}_1 \leftarrow \mathcal{DH}_\mathscr{E}(\mathsf{sk}_\mathcal{T}, \mathsf{epk}_\mathcal{C})$
20 $(k_1, k_2) \leftarrow \mathsf{KDF}(\mathsf{Z}_1, \mathsf{len}, \mathsf{info}(\mathsf{ID}_\mathcal{T}, \mathsf{epk}_\mathcal{C}))$
21 $\mathrm{cert}_\mathcal{C} \leftarrow \mathsf{AES}^{-1}_{k_1}(\mathsf{OpaqueData})$
22 if $\mathsf{C.vrf}(\mathrm{cert}_\mathcal{C}, \mathsf{pk}_{\mathcal{CA}}) = 0$ abort
23 extract $\mathsf{pk}_\mathcal{C}$ from $\mathrm{cert}_\mathcal{C}$
24 delete temporary keys $\mathsf{Z}_1, k_1$
25 $\mathsf{Z} \leftarrow \mathcal{DH}_\mathscr{E}(\mathsf{esk}_\mathcal{T}, \mathsf{pk}_\mathcal{C})$
26 $(\mathsf{sk_{cfrm}}, \mathsf{sk_{MAC}}, \mathsf{sk_{Enc}}, \mathsf{sk_{RMAC}}, \mathsf{nextOtID}, \mathsf{nextZ})$
     $\leftarrow \mathsf{KDF}(\mathsf{Z}, \mathsf{len}, \mathsf{info}($
         $\mathsf{ID}_\mathcal{T}, \mathsf{otID}_{|1..8}, \mathsf{epk}_{\mathcal{T}|1..16}, k_2$
         $))$
27 delete keys $\mathsf{Z}, k_2, \mathsf{esk}_\mathcal{T}, \mathsf{epk}_\mathcal{T}$
28 check $\mathsf{authcrypt} =$
     $\mathsf{CMAC}_{\mathsf{sk_{cfrm}}}($
         "KC_1_V"$\|\mathsf{otID}_{|1..8}\|\mathsf{ID}_\mathcal{T}\|\mathsf{epk}_{\mathcal{T}|1..16}$
     $)$
29 delete $\mathsf{sk_{cfrm}}$
```

Fig. 1. The shaded parts describe OPACITY with Full Secrecy without persistent binding. The complete protocol, as well as a line by line description is provided in the full version. The unshaded lines should give a high-level overview of the underlying Diffie-Hellman key exchange.

card's ephemeral key. Analogously, the second Diffie-Hellman key is derived from the terminal's ephemeral key $\mathsf{epk}_\mathcal{T}$ and the card's long-term key $\mathsf{pk}_\mathcal{C}$. Both keys are then used in a cascade of two key-derivation steps to derive the session keys. The card replies with its encrypted certificate (for privacy reasons), a MAC for authentication, a control byte for renegotiation, and its ephemeral public key. Assuming both parties are honest, the terminal can decrypt and validate the card's certificate, validate the MAC, and compute the session keys, too. We give the full protocol and its line-by-line description in the full version.

3.2 Preliminaries

Certificates. OPACITY uses certificates in the card verifiable certificate format (CVC) which is standardized as part of ISO 7816 — Part 8 [26] (to fully formalize our analysis, we define certification schemes in the full version of the paper). Apart from the owner's public key and an identifier for the certification authority, certificates contain application-specific data which can be used to identify the card holder. In OPACITY, this 128-bit field is called GUID and identifies the holder of the card. O-ZKM encrypts GUID using AES and the derived session key. O-FS, on the other hand, encrypts the entire certificate under an intermediate key. The (outdated) source code uses AES in CBC mode with the constant 0-vector as initialization vector. In O-FS, since the key is derived freshly upon every invocation and only used for a single encryption, this should not pose a security threat. For O-ZKM, on the other hand, the session key is used; this might compromise security.

Other functionalities used by protocols. The protocols use a key-derivation function KDF as specified in NIST SP 800-56A (§5.8.1) [41], CMAC for message authentication as specified in NIST SP 800-38B [40] (CMAC is also used as PRF in the key-derivation function) and AES-128 (no mode specified). As hash function, SHA-256 or SHA-512 are deployed. In the analysis below we model KDF through a random oracle. The injective function info is defined according to NIST SP 800-56A and prepares the input to the key-derivation function (it can be thought of the length-encoded concatenation of its input). The input to info, and therefore to the key-derivation function, contains the terminal's identity $\mathsf{ID}_{\mathcal{T}}$ (not specified in detail, but we assume that this value is globally unique and also determines the terminal's certificate $\mathrm{cert}_{\mathcal{T}}$ uniquely) and usually parts of the ephemeral keys $\mathsf{otID} = \mathsf{epk}_{\mathcal{C}}$ and $\mathsf{epk}_{\mathcal{T}}$, like the leftmost 8 or 16 bytes, $\mathsf{otID}|_{1..8}$ and $\mathsf{epk}_{\mathcal{T}}|_{1..16}$, respectively.

Security parameters. OPACITY specifies 6 parameter sets describing the length of keys and nonces, block-ciphers, and hash functions. The standard set CS2 recommends to use SHA-256 as hash function, AES-128 for encryption and MACs, and ECDH-256 for static and ephemeral keys. Nonces are 16 bytes long. By contrast, the "very strong security" setting (CS6) uses SHA-512, AES-256, ECDH-512, and 32-byte nonces. In the first case it is claimed that the resulting channel strength is 128 bits, and for CS6 the channel strength is supposedly 256.

Persistent binding. Both protocols can be run in a renegotiation mode which gives a slight performance increase if card and terminal have already successfully exchanged keys. This mode, called *persistent binding*, requires both parties to store intermediate secret values. For lack of space, we refer to the full version for a complete description of the persistent binding as well as to the analysis of the security properties in regard to this mode.

3.3 Related DH Key-Agreement Protocols

We only discuss Diffie-Hellman-based key exchange protocols which are very similar in structure to OPACITY, i.e., pairwise mix static and ephemeral Diffie-Hellman keys of the partners. These are NSA's Key Exchange Algorithm (KEA) and its variants variant KEA+ [35] and KEA+C [32]. Another closely related approach are the schemes described by ANSI X9.63 called "Unified Model" (UM) key-agreement protocols. The UM protocols have been analyzed cryptographically in [37].

Although sharing a similar skeleton —a DH key-agreement protocol using both static and ephemeral keys— the analyses of KEA, UM and their variants [35,32,37] can only serve as a very vague starting point for OPACITY; the protocols differ in numerous security-relevant details. One distinctive property of our analysis here is also that we investigate low-level details more explicitly. Considering such details makes the evaluation more delicate and complex but, on the other hand, gives a more concrete perception of the (in)security of the actual protocol. This, in particular, also concerns the renegotiation step in OPACITY which is neither supported by KEA nor by UM. Our analysis for OPACITY also needs to take additional privacy properties into account. Hence, even if OPACITY resembles the other schemes, the existing analyses provide rather weak implications for OPACITY's overall security (if any at all).

4 Security Analysis of O-FS

The concrete security parameters proposed for O-FS can be found in Section 3; however, for the sake of generality, our analysis features abstract parameters, e.g. instead of the concrete bit size of the proposed curve $\mathscr{E}$, defined on the field $\mathcal{K}$, we write $\#\mathscr{E}(\mathcal{K})$ (this is, in fact, the size of a prime-order subgroup of points). Thus, our analysis formally bounds the success probability of adversaries for *any* proposed set of parameters.

We note that the protocol itself is not perfectly correct in the sense that two honest parties may not derive the same session keys, namely, if renegotiation identifies the wrong previous connection. However, the likelihood of this event, as we detail in the full version, is in the order of $q_e^2 \cdot 2^{-128}$ for q_e executions for the recommended parameters, such that we may simply neglect such mismatches in our analysis. Nonetheless, it would be preferable to specify the behavior for this case clearly in the protocol description.

4.1 Security Assumptions

We prove O-FS secure under the elliptic curve *Gap Diffie–Hellman* (GDH) assumption [42] (by default we assume *all* hard problems are on elliptic curves, omitting to state this explicitly). Informally, the GDH assumption states that the CDH problem remains hard even when given access to an oracle $\mathsf{DDH}(\cdot,\cdot,\cdot)$, which tells whether three group elements form a Diffie–Hellman tuple or not. More formally, let $\langle G \rangle$ be an (additive) group of prime order q and generator $G \in \mathscr{E}$. The GDH problem is (t, Q, ϵ)-hard in $\langle G \rangle$ if any algorithm $\mathcal{A}$ running in time t and making at most Q queries to DDH can, on input $\langle G \rangle, G, sG, tG$, for random s, t, computes stG with probability at most ϵ. We write $\mathbf{Adv}^{\mathrm{GDH}}(t, Q)$ for (a bound on) the probability of any (t, Q)-bounded $\mathcal{A}$ solving the GDH problem.

We use standard cryptographic notation for the other involved primitives. The certification scheme $\mathsf{Cert} = (\mathsf{C.kgen}, \mathsf{C.sign}, \mathsf{C.vrf})$ is modeled as a signature scheme where the signer is a certification authority ($\mathcal{CA}$); $\mathbf{Adv}^{\mathrm{forge}}_{\mathsf{Cert}}(t, Q)$ denotes the maximal probability of forging a fresh certificate within t steps and after requesting at most Q certificates. We use $\mathbf{Adv}^{\mathrm{IND\text{-}CPA}}_{\mathsf{AES}}(t, Q)$ to denote the maximal probability of distinguishing AES ciphertexts (in CBC mode) within t steps for at most Q challenge ciphertexts (see the remark in Section 3.2 about the actual encryption mode), and $\mathbf{Adv}^{\text{forge}}_{\mathsf{CMAC}}(t, Q)$ for the maximal probability of forging a CMAC in t steps after seeing at most Q MACs. Finally, the key-derivation function (KDF) is modeled as a random oracle.

4.2 Key Secrecy and Impersonation Resistance

For the key-secrecy proof we consider sessions as indicated in Section 2, such that the session id sid for O-FS is set as $\mathsf{sid} = (\mathsf{otID}|_{1..8}, \mathsf{ID}_{\mathcal{T}}, \mathsf{epk}_{\mathcal{T}}|_{1..16})$; the partner id pid is set to the identity $\mathsf{ID}_{\mathcal{T}}$ on the card's side resp. to GUID on the terminal's side. We observe that session id's are usually preferred to comprise the entire communication transcript. The reason is that, roughly, the more information contained in sid, the "tighter" the binding of session keys to specific executions. In this sense, our formally more loose (but, according to the protocol, presumably inevitable) choice for sid's here ties executions to partners, identified via parts of the public keys and the ephemeral keys. Indeed, one easy enhancement for the protocol would be to include the card's certificate in the key-derivation step, or at least its entire public key.

The next theorem shows that O-FS is secure as a key agreement protocol, i.e., O-FS provides key secrecy.

Theorem 1 (Key Secrecy of O-FS). *In the random-oracle model,*

$$\begin{aligned}\mathbf{Adv}^{ake}_{\Pi_{\mathsf{OFS}}}(t, q_e, q_h) \le \mathbf{Adv}^{forge}_{\mathsf{Cert}}(t, q_e) + \frac{3q_e(2q_e + q_h)}{2^{\min\{\ell_{k_2}, \ell_{\mathsf{Z}}\}}} \\ + 2q_e^2 \cdot \mathbf{Adv}^{\mathrm{GDH}}(t + O(\lambda \cdot q_e \log q_e), 2q_e + q_h)\end{aligned}$$

where λ denotes the security parameter, t the running time of adversary $\mathcal{A}$, and q_e (resp. q_h) the number of executions (resp. queries to the random oracle), and ℓ_{k_2} and ℓ_{Z} denote the bit lengths of values k_2 resp. Z.

Note that key secrecy does not rely on the security of the authenticated encryption (which only enters the impersonation resistance proof), nor the secrecy of the certificate (which is only used for privacy). At the same time neither step does harm to key secrecy.

Impersonation Resistance. In this section we show that O-FS achieves impersonation resistance. Recall that this means that a malicious card cannot make an honest terminal accept, unless it is a pure relay attack and there is a card session with the same sid.

Theorem 2 (Impersonation Resistance of O-FS). *In the random-oracle model,*

$$\begin{aligned}\mathbf{Adv}^{ir}_{\Pi_{\mathsf{OFS}}}(t, q_e, q_h) \le 2q_e \cdot \mathbf{Adv}^{forge}_{CMAC}(t + O(\lambda \cdot q_e \log q_e), 0) \\ + 4q_e \cdot \mathbf{Adv}^{ake}_{\Pi_{\mathsf{OFS}}}(t, q_e, q_h)\end{aligned}$$

where λ denotes the security parameter, t the running time of adversary $\mathcal{A}$, and q_e (resp. q_h) the number of executions (resp. queries to the random oracle).

The proof follows (almost) directly from the key secrecy proof, noting that in order to be able to impersonate one would need to compute a MAC for the secure key $\mathsf{sk}_{\mathsf{cfrm}}$.

4.3 Privacy

Though O-FS does not attain untraceability, it does, nevertheless, provide identity hiding. This holds as long as we assume that the unspecified mode of encryption of $\mathrm{cert}_{\mathcal{C}}$ with AES is secure (see our remark in Section 3.2).

Theorem 3 (Identity-Hiding in O-FS). *In the random-oracle model,*

$$\begin{aligned}\mathbf{Adv}^{\textit{id-hide}}_{\Pi_{\mathsf{OFS}}}(t, q_e, q_t, q_h) \leq \quad & \frac{1}{2} + \mathbf{Adv}^{\textit{forge}}_{\mathsf{Cert}}(t, q_e) + \frac{2q_t(2q_t + q_h)}{2^{\ell_{k_2}}} \\ & +q_e^2 \cdot \mathbf{Adv}^{\mathsf{GDH}}(t + O(\lambda \cdot q_e \log q_e), 2q_e + q_h) \\ & + \frac{q_e q_t}{\#\mathscr{E}(\mathcal{K})} + q_t \cdot \mathbf{Adv}^{\textit{IND-CPA}}_{\mathsf{AES}}(t + O(q_t)) \ .\end{aligned}$$

where λ denotes the security parameter, t the running time of the adversary, and q_e (resp. q_t, q_h) the number of executions (resp. Test*-sessions and queries to the random oracle).*

5 Security of the Channel Protocol

Here we discuss briefly the security of the secure messaging (used both in ZKM and FS) and of the composition of the channel with the key agreement step.

Secure Messaging. Once the keys are generated the parties use them to secure the communication. The description [24] proposes two modes, one for command and response MACs without confidentiality (using keys $\mathsf{sk}_{\mathsf{MAC}}$ and $\mathsf{sk}_{\mathsf{RMAC}}$, respectively), and the other one for encrypted data transfer under the key $\mathsf{sk}_{\mathsf{Enc}}$ used by both parties. If only authenticity is required, then the data is secured according to ISO 7816-4 [25]; in case encryption is used the protocol basically follows the encrypt-then-MAC approach, first encrypting the payload.

Alarmingly, according to the standard [24], the terminal can ask the card via the control byte to only create a single key $\mathsf{sk}_{\mathsf{Enc}} = \mathsf{sk}_{\mathsf{RMAC}} = \mathsf{sk}_{\mathsf{MAC}}$, operating in a special mode (ONE_SK). Sharing the key among different primitives usually needs a cautionary treatment. It remains unclear why OPACITY implements this mode, but it does not seem to be recommendable from a pure security perspective. In what follows we assume that independent keys are used instead.

Encryption for the encrypt-then-MAC approach in the secure messaging is not further specified in [24]. The (outdated) implementation relies on AES encryption with a prepended, fixed-length session counter. For authentication the parties first pad the message (or ciphertext) according to ISO 7816-4, basically prepending a MAC chaining value of 16 bytes before computing an AES-based CMAC [7,28] according to SP 800-38B [40]. We omit a formal analysis of secure messaging which, except for the single-key mode, follows the common cryptographic approaches. It would be nonetheless interesting to provide such an analysis, taking into account recent attacks and models for such steps [31,2,45,46]. However, it is beyond our scope here.

Composition. Clearly, a secure key-exchange protocol and secure messaging on their own may not be enough to ensure the security of the composed protocol. Several approaches exist to bridge this gap, ranging from monolithic analysis of the composed protocol, to general-purpose compositional frameworks like Canetti's Universal Composition (UC) model [13]. The latter has been successfully applied to analyze and construct securely composable key-exchange protocols [14]. However, security of key exchange in the UC model (and comparable simulation-based frameworks [33]) already imposes strong requirements on the protocols which are hard to meet.

Since we analyzed O-FS in the game-based BR-model we can apply the recent result by Brzuska et al. [12] to conclude overall channel security of the key agreement combined with secure messaging. This holds as long as O-FS provides a property called public session matching [12], which we discuss in the full version to be true. Since we do not recommend to use O-ZKM we do not address the question for this protocol.

6 Conclusion

Our analysis reveals that, from a cryptographic point of view, O-FS achieves a decent level of key secrecy, but has clear restrictions on privacy guarantees. For one, privacy could be improved by also encrypting the card's control byte $\mathtt{CB}_{\mathcal{C}}$ for persistent binding, hiding the fact if the card has been used in connection with that terminal before. Whereas the situation for O-FS is arguable, we do not recommend O-ZKM for deployment. This is due to its rather weak security guarantees for (terminal) authentication and the weaker form of identity hiding.

Our analysis also shows common problems in making precise security claims about real protocols. Like with every cryptographic (or scientific) model we have to abstract out some details. This can be an impediment in particular in view of the fact that the protocol can operate in various modes, e.g., for compatibility reasons. This complexity is the cryptographer's enemy, discussing all possibilities is often beyond a reasonable approach. However, omitting some of these modes is dangerous, as they often admit back doors for attackers. There are some *potential* back doors for OPACITY as well, e.g., the single-key mode ONE_SK for secure messaging. This is magnified by the fact that OPACITY is not fully specified with respect to all relevant details (e.g., which encryption mode is used for OpaqueData). Also, the binding of sessions to their keys is rather loose as merely a partial transcript of the execution enters the key derivation resp. message authentication. In this sense, it should be understood that our (partly positive) cryptographic analysis has its inherent limitations.

Acknowledgments. We thank the anonymous reviewers for comments. Marc Fischlin was supported by a Heisenberg grant Fi 940/3-1 of the German Research Foundation (DFG). This work was supported by CASED (`www.cased.de`) and by EC SPRIDE.

References

1. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
2. Albrecht, M.R., Paterson, K.G., Watson, G.J.: Plaintext recovery attacks against SSH. In: 2009 IEEE Symposium on Security and Privacy, pp. 16–26. IEEE Computer Society Press (May 2009)
3. Smart Card Alliance: Industry technical contributions: Opacity (April 2013), `http://www.smartcardalliance.org/pages/smart-cards-contributions-opacity`
4. ANSI X9-63-199X – Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography (1999)
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
6. Bender, J., Dagdelen, Ö., Fischlin, M., Kügler, D.: Domain-specific pseudonymous signatures for the german identity card. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 104–119. Springer, Heidelberg (2012)
7. Black, J., Rogaway, P.: CBC MACs for arbitrary-length messages: The three-key constructions. Journal of Cryptology 18(2), 111–131 (2005)
8. Boyd, C., Mao, W., Paterson, K.G.: Deniable authenticated key establishment for internet protocols. In: Security Protocols Workshop, pp. 255–271 (2003)
9. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy. The MIT Press (2000)
10. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 132–145. ACM Press (October 2004)
11. Brzuska, C., Fischlin, M., Smart, N., Warinschi, B., Williams, S.: Less is more: Relaxed yet composable security notions for key exchange. Cryptology ePrint Archive, Report 2012/242 (2012), `http://eprint.iacr.org/`
12. Brzuska, C., Fischlin, M., Warinschi, B., Williams, S.C.: Composability of Bellare-Rogaway key exchange protocols. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS 2011, pp. 51–62. ACM Press (October 2011)
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press (October 2001)
14. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
15. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM 28 (October 1985)
16. Dagdelen, Ö., Fischlin, M., Gagliardoni, T., Marson, G.A., Mittelbach, A., Onete, C.: A cryptographic analysis of OPACITY. Cryptology ePrint Archive, Report 2013/234 (2013), `http://eprint.iacr.org/`

17. Deeg, M., Eichelmann, C., Schreiber, S.: Programmed insecurity — SySS cracks yet another usb flash drive, `http://www.syss.de/fileadmin/ressources/040_veroeffentlichungen/dokumente/SySS_Cracks_Yet_Another_USB_Flash_Drive.pdf`
18. Deeg, M., Schreiber, S.: Cryptographically secure? SySS cracks a usb flash drive, `https://www.syss.de/fileadmin/ressources/040_veroeffentlichungen/dokumente/SySS_Cracks_SanDisk_USB_Flash_Drive.pdf`
19. Di Raimondo, M., Gennaro, R.: New approaches for deniable authentication. Journal of Cryptology 22(4), 572–615 (2009)
20. Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and on-line deniability of authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Heidelberg (2009)
21. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM 51(6), 851–898 (2004)
22. Gajek, S., Manulis, M., Pereira, O., Sadeghi, A.-R., Schwenk, J.: Universally composable security analysis of TLS. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 313–327. Springer, Heidelberg (2008)
23. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Des. Codes Cryptography 67(2), 245–269 (2013)
24. INCITS: 504-1, Information Technology - generic identity command set part 1: Card application command set
25. ISO/IEC: Identification cards - Integrated circuit(s) cards with contacts - Part 4: Organization, security and commands for interchange. Tech. Rep. ISO/IEC 7816-4, International Organization for Standardization, Geneva, Switzerland (2005)
26. ISO/IEC: Identification cards - Integrated circuit(s) cards with contacts - Part 8: Security related interindustry commands. Tech. Rep. ISO/IEC 7816-8, International Organization for Standardization, Geneva, Switzerland (2009)
27. ISO/IEC: Identification Cards – Integrated Circuit Cards Programming Interface – Part 6: Registration procedures for the authentication protocols for interoperability. Tech. Rep. ISO/IEC FDIS 24727-6, International Organization for Standardization, Geneva, Switzerland (2009)
28. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
29. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (2012)
30. Juels, A., Weis, S.A.: Defining strong privacy for RFID. Cryptology ePrint Archive, Report 2006/137 (2006), `http://eprint.iacr.org/`
31. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
32. Kudla, C., Paterson, K.G.: Modular security proofs for key agreement protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer, Heidelberg (2005)
33. Küsters, R., Tuengerthal, M.: Composition theorems without pre-established session identifiers. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS 2011, pp. 41–50. ACM Press (October 2011)
34. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)

35. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
36. Le, T.V., Burmester, M., de Medeiros, B.: Universally composable and forward-secure RFID authentication and authenticated key exchange. In: Bao, F., Miller, S. (eds.) ASIACCS 2007, pp. 242–252. ACM Press (March 2007)
37. Menezes, A., Ustaoglu, B.: Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard. In: Abe, M., Gligor, V. (eds.) ASIACCS 2008, pp. 261–270. ACM Press (March 2008)
38. Morrissey, P., Smart, N.P., Warinschi, B.: The TLS handshake protocol: A modular analysis. Journal of Cryptology 23(2), 187–223 (2010)
39. NIST: Security Requirements for Cryptographic Modules. Tech. Rep. FIPS 140-2, National Institute of Standards and Technology (2002)
40. NIST: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Tech. Rep. SP 800-38B, National Institute of Standards and Technology (2007)
41. NIST: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. Tech. Rep. SP800-56A, National Institute of Standards and Technology (2007)
42. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
43. OPACITY: Reference Implementation - `sourceforge.net/projects/opacity/`
44. Ouafi, K., Phan, R.C.-W.: Privacy of recent RFID authentication protocols. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 263–277. Springer, Heidelberg (2008)
45. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag size *does* matter: Attacks and proofs for the TLS record protocol. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 372–389. Springer, Heidelberg (2011)
46. Paterson, K.G., Watson, G.J.: Authenticated-encryption with padding: A formal security treatment. In: Naccache, D. (ed.) Cryphtography and Security: From Theory to Applications. LNCS, vol. 6805, pp. 83–107. Springer, Heidelberg (2012)
47. Saint, E.L., Fedronic, D.L.J.: Open protocol for authentication and key establishment with privacy (July 2010)
48. Saint, E.L.: Opacity - the new open protocol of choice (August 2012), `http://www.itsecurityhub.eu/2012/08/opacity-the-new-open-protocol-of-choice/`
49. Saint, E.L.: Personal communication (July 2012)
50. Saint, E.L., Fedronic, D., Liu, S.: Open protocol for access control identification and ticketing with privacy (July 2011), `http://www.smartcardalliance.org/resources/pdf/OPACITY_Protocol_3.7.pdf`
51. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 55–69. Springer, Heidelberg (2009)