A Closer Look at PKI: Security and Efficiency

Alexandra Boldyreva[1] and Marc Fischlin[2] and Adriana Palacio[3] and Bogdan Warinschi[4]

[1]Georgia Institute of Technology, USA. sasha@gatech.edu

[2]Darmstadt University of Technology, Germany. marc.fischlin@gmail.com

[3]Bowdoin College, USA. apalacio@bowdoin.edu

[4]University of Bristol, UK. bogdan@cs.bris.ac.uk

April 13, 2007

Abstract

In this paper we take a closer look at the security and efficiency of public-key encryption and signature schemes in public-key infrastructures (PKI). Unlike traditional analyses which assume an "ideal" implementation of the PKI, we focus on the security of joint constructions that consider the certification authority (CA) and the users, and include a key-registration protocol and the algorithms of an encryption or a signature scheme. We therefore consider significantly broader adversarial capabilities. Our analysis clarifies and validates several crucial aspects such as the amount of trust put in the CA, the necessity and specifics of proofs of possession of secret keys, and the security of the basic primitives in this more complex setting. We also provide constructions for encryption and signature schemes that provably satisfy our strong security definitions and are more efficient than the corresponding traditional constructions that assume a digital certificate issued by the CA must be verified whenever a public key is used. Our results address some important aspects for the design and standardization of PKIs, as targeted for example in the standards project ANSI X9.109.

1 Introduction

Public key cryptography implicitly relies on the existence of a *public-key infrastructure* (PKI), where each user has a pair of public and secret keys for the cryptosystem, and that this association is publicly available. The designers of public-key cryptosystems always define how the public and the secret keys are generated and used, but almost never carefully specify how the binding between keys and user identities takes place. The tacit assumption is that this binding is established *a priori* through PKI management operations.

1.1 Motivation

The policies and the procedures regarding PKIs are continuously changing and detailed descriptions are invariably long and tedious.[1] Unfortunately, existing literature still does not answer several important questions.

[1]See for example the document that describe the current state-of-the-art: "Internet X.509 Public Key Infrastructure – Certificate Management Protocol (CMP)" [1].

What exactly is the certification authority (CA), the entity that links public keys to identities, trusted not to do? Can and should some degree of security be ensured even when the CA is malicious or becomes compromised? Proofs of possession (POP) —in which a user proves possession of the secret key when registering a public key with the CA— are a defense mechanism for protecting against rogue-key and key-substitution attacks, but what exactly should they be and, more importantly, are they really necessary?

A question that is perhaps even more important is whether provably-secure encryption and signature schemes are indeed secure when used in a particular PKI. Although it is largely believed to be the case, the question is far from moot since most existing schemes are analyzed in settings where compositional aspects are neglected. In particular, the security of the combination of a key-registration protocol with existing encryption or signature schemes does not immediately follow from the security of the individual components. In principle, by cleverly combining its ability to attack the key-registration protocol and its ability to attack the primitive (encryption or signatures), an adversary could mount a successful attack against the joint construction.

Limitations of security analyses that do not explicitly include the behavior of the CA or the key-registration protocol have been previously pointed out in other contexts. In the case of key exchange, Shoup [40] suggests that registration of public keys should be considered explicitly as part of the key agreement protocol to be analyzed. Kaliski [26] exemplifies the importance of such measures by presenting unknown key-share attacks on the MQV key exchange protocol [33]. These attacks could have been discovered with a thorough analysis that considers the CA as an active party participating in the protocol. We review further related work at the end in Section 6.

1.2 Contributions

In this paper we initiate a study of PKIs with respect to security of the two most important public-key primitives: encryption and digital signature schemes. Our main motivation is to answer the questions raised above and other related issues.

Models. Security arguments in the absence of rigorous models do not provide strong security guarantees, and such models are conspicuously absent in the case of PKIs. Our first contribution are rigorous definitions for primitives when used in this setting together with appropriate security notions. The inherent complexity of the PKI settings, the non-typical adversarial powers, and the difficulty of precisely identifying the situations that constitute a security breach make the design of such models an entirely non-trivial task.

Since security goals depend on the primitive used, we treat the cases in which keys are used for encryption and for signing separately. Specifically, we define two primitives, called *certified encryption* and *certified signature* schemes, and for each primitive we define a notion of security. Besides the standard algorithms for encryption and signing, we model explicitly interactive protocols for registering the public keys with a CA. Consequently, our security notions are against an adversary with broad capabilities that take into account threats arising from the key-registration protocol, possibly run concurrently, the presence of several parties, including the users and the (possibly corrupt) CA. The details are in Sections 2, 3 and 4.

Our security definitions are general and powerful. The models we propose directly capture settings where users have multiple public keys, and where keys have additional attributes, such as an expiration date. They easily extend to handle hierarchical certification and certificate revocation. Moreover, while we capture the original goal for which PKI was invented we make flexible assumptions on how certification is achieved. In particular, schemes that aim at achieving certification but avoid the original mechanism of explicit certificates specific to the traditional PKIs (e.g. schemes similar to those in [20, 2]) can still be analyzed in our models. We provide a detailed discussion in Sections 3 and 4.

The design of our models in general and that of the security goals in particular are motivated by the "core" properties of the primitives, namely, confidentiality for encryption and integrity and authenticity for signatures. For protocols in which encryption schemes or signatures are used beyond these basic properties, e.g., encryption schemes used as commitments, additional analysis in light of the new goals is required. Yet, our attack model should be easily transferable to those scenarios, and only the security definitions would need to be adapted.

Analysis of Traditional Schemes. Next we focus on constructions that satisfy the proposed notions of security. We start with an analysis of "traditional" certified encryption and certified signature schemes. In these constructions, the CA uses a signature scheme to issue digital certificates, and then parties produce ciphertexts (resp., signatures) using a standard encryption (resp., signature) scheme. These schemes are defined in detail in Sections 3 and 4, respectively.

Although it seems folklore that the traditional approach is "secure", to the best of our knowledge no formal validation in a sound model with respect to clearly expressed security goals has been devised prior to our work. We offer a rigorous analysis that shows that these schemes are indeed secure in the appropriate security model we design. Our proof gives concrete security bounds that support recommendations for practical parameter choices. While expected, these results are important to increase confidence in the use of the schemes and allow to make security statements based on solid foundations. Our concrete security results are in Sections 3 and 4.

The results that we obtain regarding the design of proofs of possession are less expected, if not surprising. Our investigation shows that formal proofs of knowledge are not necessary for basic security of the certified encryption and signature schemes, and that simpler challenge-response protocols suffice. For signatures, the user simply signs a distinct message[2] provided by the CA. Perhaps surprisingly, we show that for basic encryption no proof of possession is required. Intuitively, in the case of encryption, this means that data privacy is not compromised if a user does not have the secret key associated to the public key it registers. We note that these results do not eliminate the proof-of-knowledge requirements imposed on these primitives in other settings (e.g., [4, 10, 9, 23, 31, 35]) and only concern the security of certified encryption and signatures.

More efficient constructions. Since our models do not require that solutions use explicit certificates as in the traditional constructions, it is natural to ask if it is possible to obtain improvements over the traditional solutions, e.g., in terms of efficiency. We answer this question affirmatively. We present more efficient constructions for certified encryption and certified signature schemes that use implicit certificates therefore avoiding the explicit verification of the binding between public keys and identities.

Our certified encryption scheme uses a variant of ElGamal encryption [18] combined with implicit certificates realized through Schnorr signatures, and is proven secure according to our definition in the random oracle model [6] under the Computational Diffie-Hellman assumption. This scheme is more efficient than the traditional certified encryption scheme where the CA uses Schnorr signatures to issue explicit certificates and users employ ElGamal encryption[3]. For security parameter k the latter requires $4.75k$ modular multiplications to encrypt (using the square-and-multiply exponentiation method combined with well-known speed-up techniques for multi-exponentiations) while our scheme only requires $3.25k$ multiplications, coming thus quite close to the performance of regular ElGamal encryption without certification.

For signatures, we propose a construction based on Schnorr signatures [37], provably secure according to our definition in the random oracle model under the Discrete Logarithm assumption. Compared to the traditional approach of using such signatures as explicit certificates, our solution reduces the average number of modular multiplications for verification from $3.5k$ to $1.875k$, and thus achieves almost the same efficiency as regular Schnorr signatures without certification. Notice that the increase in efficiency comes at the expense of a loss in provable security due to looser reductions. It is an open problem to find tighter reductions.

We note that in the stateful settings where valid certificates of the other parties are stored permanently, traditional schemes are the expedient choice. For the stateless case, however, our constructions offer computational savings over the traditional approach.

2 Modeling Public-Key Infrastructures

To model public-key infrastructures we assume that there is a designated party, the certification authority (CA), and a set of users. Each user has a unique identity $\mathsf{ID} \in \{0,1\}^*$ in form of an X.509 entry, an e-mail address or a similar distinguished name. The identity may also contain auxiliary information like an expiration date which refers for example to the contract period of an employee or to the validity period of the certificate.

Certification Authority. The CA holds a public key pk_{CA} and a corresponding secret key sk_{CA}. We presume that the public key is authenticated and known to all parties, i.e., once it is published it cannot be changed by the adversary. This is usually accomplished by a hierarchical arrangement of CAs, each intermediate CA certifying the validity of the public key of its successor. Only the key of the root CA has to

[2]It is necessary to ensure that this message will not be signed by this user later. One way to achieve this, which is also our approach, is to prepend the "challenge" messages chosen by the CA with `0`, and the messages the user signs with `1`.

[3]Or a version of ElGamal that is IND-CCA secure in the random oracle model.

be authenticated by other means. Here we focus on the the simpler one-tier approach of having only one CA, i.e., our model can be viewed as a condensed hierarchy with our single CA as the root CA. We discuss the more general case of hierarchical CAs in Section 5.2.

Registration of Keys. Each user can register keys with the CA by running the registration protocol. The required validation of the user's identity ID is usually done before by the so-called registration authority (RA), which sometimes coincides with the CA. Checking the identity of the user wishing to register its public key is typically performed by the RA through personal identification and physical validation (e.g., with help of a passport or a driver license). Hence, this part is beyond our computational model and we simply assume that bindings between user identities and their public keys are authentic.

We do not assume the existence of private channels. We do, however, presume authenticated channels between the CA and the users, even though the user most likely does not have a certified signature key when the registration starts. Without this minimal assumption about authenticated communication achieving any reasonable security guarantee seems to be impossible. The assumption can be enforced by a variety of means that include for example having the certification authority confirm the registration of a key through regular mail, signed electronic mail (with the signature verification key included in pk_{CA}), legally binding documents, or simply meeting in person.

The registration protocol itself is defined very generically. In this process the user derives a public key pk which may be used for encryption or signature verification and a secret key sk for decrypting or signing. We do not specify how the keys are generated (i.e., picked by the user alone or generated jointly between the user and the CA), yet we postulate that the CA should not be able to learn the corresponding secret key of the user. This inevitably requires interaction between both parties. The user also obtains a certificate cert which, classically, is an *explicit certificate* of type X.509, including the CA's signature. But since we also use other approaches like *implicit certificates* cert should be rather thought of as an arbitrary, possibly empty string. We assume, however, that each pair (ID, pk), where pk is registered, is unique; this can be achieved as is done for X.509 certificates by issuing serial numbers or other auxiliary information.

Revocation. For simplicity, we do not introduce revocation techniques in our basic model. We further discuss how to augment our definitions and schemes to address revocations in Section 5.3.

3 Secure Encryption in Public-Key Infrastructures

Syntax of Certified Encryption Schemes. A *certified encryption scheme* is a tuple $\mathsf{CS} = (\mathcal{EG}, \mathcal{K}, (\mathcal{C}, \mathcal{U}), \mathcal{E}, \mathcal{D})$ of probabilistic polynomial-time algorithms:

- $\mathcal{EG}$ is a randomized *parameter-generation* algorithm. It takes input 1^k, where k is the security parameter, and outputs some global parameters I, available to all parties. For sake of readability we omit I from the input of the parties.
- $\mathcal{K}$ is a randomized *key-generation* algorithm. It takes input I, and outputs a pair $(pk_{\mathsf{CA}}, sk_{\mathsf{CA}})$ consisting of a public key and a matching secret key.
- $(\mathcal{C}, \mathcal{U})$ is a pair of interactive randomized algorithms forming the (two-party) *public-key registration protocol*. $\mathcal{C}$ takes input a secret key sk_{CA}. $\mathcal{U}$ takes input the identity ID of a user and the public key pk_{CA} corresponding to sk_{CA}. As result of the interaction, the output of $\mathcal{C}$ is $(\mathsf{ID}, pk, \mathsf{cert})$, where pk is a public key and cert is an issued certificate. The local output of $\mathcal{U}$ is $(\mathsf{ID}, pk, sk, \mathsf{cert})$, where sk is a secret key that user ID uses to decrypt ciphertexts. We write $((\mathsf{ID}, pk, \mathsf{cert}), (\mathsf{ID}, pk, sk, \mathsf{cert})) \xleftarrow{\$} (\mathcal{C}(sk_{\mathsf{CA}}), \mathcal{U}(\mathsf{ID}, pk_{\mathsf{CA}}))$ for the result of this interaction. Either party can quit the execution prematurely, in which case the output of the party is set to $\perp$.
- $\mathcal{E}$ is a randomized *encryption* algorithm that takes input a user's identity ID, a public encryption key pk, a certificate cert, the authority's public key pk_{CA}, and a message $M \in \mathrm{MsgSp}(I)$, and outputs a ciphertext $C \in \{0,1\}^* \cup \{\perp\}$.
- $\mathcal{D}$ is a deterministic *decryption* algorithm which takes input a user's identity ID, a secret decryption key sk, a certificate cert, the authority's public key pk_{CA}, and a ciphertext C, and outputs $M \in \mathrm{MsgSp}(I) \cup \{\perp\}$. If $M = \perp$ we say that the ciphertext C is invalid (relative to $\mathsf{ID}, sk, \mathsf{cert}, pk_{\mathsf{CA}}$).

The scheme is *correct* iff for any parameters I, any pk_{CA}, any message $M \in \mathrm{MsgSp}(I)$, any user ID, and any $((\mathsf{ID}, pk, \mathsf{cert}),\ (\mathsf{ID}, pk, sk, \mathsf{cert})) \xleftarrow{\$} (\mathsf{CA}(sk_{\mathsf{CA}}),\ \mathcal{U}(\mathsf{ID}, pk_{\mathsf{CA}}))$, and any $C \xleftarrow{\$} \mathcal{E}(\mathsf{ID}, pk, \mathsf{cert}, pk_{\mathsf{CA}}, M)]$, it holds that $\mathcal{D}(\mathsf{ID}, sk,\ pk_{\mathsf{CA}}, \mathsf{cert}, C) = M$.

Remark 1. Our syntax does not explicitly deal with verifying the certificates, even though this may be necessary for security of the scheme. We assume that the constructions include such checks as part of their encryption algorithms.

Remark 2. The certificateless encryption schemes of [20, 2] are special cases of certified encryption schemes where the certificate cert is empty.

Security of Certified Encryption Schemes. We start with an informal discussion of the more interesting aspects of our model for secure certified encryption, and motivate some of the design choices that we made.

We envision a powerful adversary that is allowed to even corrupt the CA (i.e. learn its secret key and act on its behalf). At a superficial glance it may seem that no security requirements would make sense in this case since under these circumstances the adversary could create new keys with valid certificates on behalf of honest users, and then decrypt any ciphertext created with these keys. We wish however to ensure that even if the CA is corrupt, the communication encrypted with keys truly registered by honest users is still protected. At least that requires the CA not to have users' secret keys. This requirement is somewhat akin to forward security. Without loss of generality, we treat the case when the corruption of the CA is static, i.e., the adversary decides at the beginning of its execution whether to control the CA or not. Indeed, we are able to show that our definition is equivalent (up to a constant factor in the security statement) to the analogous definition where the adversary can corrupt the CA at any point (see Section 5.1).

Naturally the fundamental security requirement for certified encryption is privacy of encrypted data. However, as discussed in the introduction, we take into account potential threats arising from the use of the registration protocol. In particular, we require that an adversary cannot pass as genuine (registered) an unregistered key upon an honest user, in a way that allows the adversary to recover messages encrypted with this key. In other words, encryptions with unregistered keys can not be decrypted by the adversary.

Our model uses the standard definitional idea of indistinguishabiliy [21] captured via left-right encryption oracles [5]. The left-right encryption oracle is initialized with a secret bit b and encrypts either the left message M_0 or the right message M_1 of the two messages submitted by the adversary. The oracle is universal in the sense that the adversary can query it about any party ID and for any (not necessarily valid) key/certificate pair pk, cert. We restrict the kind of queries that are allowed in order to exclude trivial attacks. We demand that either (1) user ID is honest and $(\mathsf{ID}, pk, \mathsf{cert})$ has been registered before with the CA, or (2) $(\mathsf{ID}, pk, \mathsf{cert})$ is not registered but the CA is still honest.

The first condition covers the case of "standard" queries for proper keys of honest users, and encompasses the case when the CA might be corrupt. The second restriction prevents the adversary to register a key for some honest user (after corrupting the CA) and to determine the bit b easily. Also, if the CA is corrupt then the adversary can generate a certificate for any user locally, without invoking the registration protocol. This would also allow the adversary to create unregistered keys for which the oracle produces a valid ciphertext and which the adversary can still decrypt. Hence, we only permit queries where the key of the user has not been registered with the honest CA.

Finally, we emphasize that in our model we do not assume that the communication between the users and the CA is encrypted, i.e., we assume public channels. We therefore avoid the "chicken-and-egg"-like problem: how to assume secret transmissions if one is still trying to establish a public encryption key through this communication?

Definition 3.1 ***[Security of Certified Encryption Schemes]*** *Let* $\mathsf{CE} = (\mathcal{G}, \mathcal{K}, (\mathcal{C}, \mathcal{U}), \mathcal{E}, \mathcal{D})$ *be a certified encryption scheme. We associate to scheme* CE*, an adversary* $\mathcal{A}$*, and a bit* b *the experiments* $\mathbf{Exp}^{\text{cenc-ind-atk}}_{\mathsf{CE},\mathcal{A},b}(k)$ *for* $\text{atk} \in \{\text{cpa}, \text{cca}\}$*. In both experiments* $\mathcal{A}$ *is given as input* $I \xleftarrow{\$} \mathcal{G}(1^k)$*. The experiment maintains two virtual arrays RegListPub, RegListSec used to store public and secret information pertaining to users (respectively). We note that* $\mathcal{A}$ *knows the elements of RegListPub but not those of RegListSec. Also the adversary has access to all transcripts of the protocols executed during the experiment.*

- Corruption of certification authority: *First,* $\mathcal{A}$ *decides if to corrupt the* CA*. If so,* $\mathcal{A}$ *chooses the key* $\mathrm{pk}_{\mathsf{CA}}$ *of the CA, else* $\mathrm{pk}_{\mathsf{CA}}$ *is generated via* $(pk_{\mathsf{CA}}, sk_{\mathsf{CA}}) \xleftarrow{\$} \mathcal{K}(I)$ *and given to* $\mathcal{A}$*.*

- Registering keys of users: *During the experiment, $\mathcal{A}$ can specify a user* ID *from the set of identities, to initiate a run of the public-key registration protocol with the honest or corrupt certification authority. If this is the first time the user* ID *is activated then $\mathcal{A}$ first decides whether to corrupt this user or not. In the execution with the CA we assume wlog. that at least one party is honest. At the end of the execution, when $\mathcal{C}$ outputs values* $(\mathsf{ID}, pk, \mathsf{cert})$ *and $\mathcal{U}$ outputs (possibly different) values* $(\mathsf{ID}', pk', sk', \mathsf{cert}')$*, we store* $(\mathsf{ID}', pk', \mathsf{cert}')$ *in RegListPub and* $(\mathsf{ID}', pk', sk', \mathsf{cert}')$ *in RegListSec if $\mathcal{U}$ is honest, or merely* $(\mathsf{ID}, pk, \mathsf{cert})$ *in RegListPub if only $\mathcal{C}$ is honest. If one of the parties is dishonest or stops prematurely then $\perp$ is stored in the corresponding array. Notice that all steps in the experiment, including steps of this interactive protocol may be arbitrarily interleaved.*
- Encryption queries: *$\mathcal{A}$ can query $\mathcal{UE}_{\mathsf{CE}}(b, pk_{\mathsf{CA}}, \cdot, \cdot, \cdot)$, a universal left-right encryption oracle. It takes as input a tuple* $(\mathsf{ID}, pk, \mathsf{cert})$ *and two messages $M_0, M_1 \in \mathrm{MsgSp}(I)$ of equal length and returns a ciphertext $C \xleftarrow{\$} \mathcal{E}(\mathsf{ID}, pk, \mathsf{cert}, pk_{\mathsf{CA}}, M_b)$. We impose the restriction that user* ID *is honest and at this point* $(\mathsf{ID}, pk, \mathsf{cert})$ *is listed in RegListPub, or that the certification authority is still honest but* $(\mathsf{ID}, pk, \mathsf{cert})$ *does not appear in RegListPub at this point.*
- Decryption queries: *In experiment $\mathbf{Exp}^{\text{cenc-ind-cca}}_{\mathsf{CE},\mathcal{A},b}(k)$ the adversary is also given access to a universal decryption oracle $\mathcal{UD}_{\mathsf{CE}}(pk_{\mathsf{CA}}, \cdots)$ which has access to the array RegListSec. The queries to the oracle are tuples* $(\mathsf{ID}, pk, \mathsf{cert}, C)$ *where we require that C has not been previously returned by oracle $\mathcal{UE}_{\mathsf{CE}}(b, pk_{\mathsf{CA}}, \cdots)$ as answer to some query* $((\mathsf{ID}, pk, \mathsf{cert}), M_0, M_1)$*. If* $(\mathsf{ID}, pk, sk, \mathsf{cert})$ *occurs in RegListSec the oracle returns $\mathcal{D}(\mathsf{ID}, sk, \mathsf{cert}, pk_{\mathsf{CA}}, C)$; otherwise, it returns $?\perp$.*

The adversary eventually stops and outputs a guess bit d which is also considered to be the output of the experiment. For $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$ *the adversary's advantages in attacking the scheme are defined as follows.*

$$\mathbf{Adv}^{\text{cenc-ind-atk}}_{\mathsf{CE},\mathcal{A}}(k) \quad = \quad \Pr[\mathbf{Exp}^{\text{cenc-ind-atk}}_{\mathsf{CE},\mathcal{A},1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{cenc-ind-atk}}_{\mathsf{CE},\mathcal{A},0}(k) = 1] \, .$$

CE *is said to be IND-CPA (resp. IND-CCA) secure if the corresponding advantage of any* $\mathrm{poly}(k)$*-time adversary $\mathcal{A}$ is negligible.* ▌

"Traditional" Certified Encryption Schemes. We confirm that the classical approach of using signature-based certificates for the encryption scheme yields a secure certified encryption scheme. We show this to be the case even when during a public key registration a user does not prove that it knows the corresponding secret key. The schemes that we use in our construction satisfy standard security notions (see the full version Section A for precise definitions of syntax and security). We now give the scheme (Construction 3.2) and state our security result (Theorem 3.3). The proof is in Appendix B.

Construction 3.2 (Traditional Certified Encryption Scheme) *Let* $\mathsf{DS} = (\mathcal{SG}_s, \mathcal{SK}_s, \mathcal{S}_s, \mathcal{V}_s)$ *be a digital signature scheme, and* $\mathsf{AE} = (\mathcal{EG}_e, \mathcal{EK}_e, \mathcal{E}_e, \mathcal{D}_e)$ *be an asymmetric encryption scheme. Define* $\mathsf{TCE} = (\mathcal{EG}, \mathcal{K}, (\mathcal{C},\mathcal{U}), \mathcal{E}, \mathcal{D})$*:*

- Parameter generation: *Algorithm $\mathcal{EG}(1^k)$ executes $I_s \xleftarrow{\$} \mathcal{SG}_s(1^k)$, $I_e \xleftarrow{\$} \mathcal{EG}_e(1^k)$ and outputs $I = (I_s, I_e)$.*
- Key generation: *Algorithm $\mathcal{K}$ generates a key pair $(pk_{\mathsf{CA}}, sk_{\mathsf{CA}}) \xleftarrow{\$} \mathcal{SK}_s(I_s)$.*
- Registration: *In order to register a key user,* ID *first generates a key pair $(pk, sk) \xleftarrow{\$} \mathcal{EK}_e(I_e)$ and sends* (ID, pk) *to $\mathcal{C}$ who computes $s \xleftarrow{\$} \mathcal{S}_s(sk_{\mathsf{CA}}, \mathsf{ID}||pk)$ and outputs* (ID, pk, s)*. The user sets* $\mathsf{cert} = s$ *and outputs* $(\mathsf{ID}, pk, sk, \mathsf{cert})$*.*
- Encryption: *To encrypt a message M under identity* ID*, public key pk, certificate* cert *and key pk_{CA} the encryption algorithm $\mathcal{E}$ first verifies with $\mathcal{V}_s$ that* cert *is a valid signature for* $\mathsf{ID}||pk$ *under key pk_{CA}. If not then return $\perp$. Else compute $C \xleftarrow{\$} \mathcal{E}_e(pk, M)$ and return C.*
- Decryption: *To decrypt a ciphertext C with* $(\mathsf{ID}, sk, \mathsf{cert})$ *and pk_{CA} run algorithm $\mathcal{D}_e(sk, C)$ and return the answer.* ▌

Theorem 3.3 *Let* DS *be a secure signature scheme and let* AE *be an IND-CPA secure (resp. IND-CCA secure) encryption scheme. Then the certified encryption scheme in Construction 3.2 is IND-CPA secure (resp. IND-CCA secure).* ▌

More precisely, for adversary $\mathcal{A}$ let Q_R be the maximal number of (initiated) registration executions during experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},0}(k)$ or $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k)$. Then there exists algorithms $\mathcal{B}_{\mathsf{DS}}$ and $\mathcal{B}_{\mathsf{AE}}$ such that for atk $\in \{\text{cpa}, \text{cca}\}$,

$$\mathbf{Adv}^{\text{cenc-ind-atk}}_{\mathsf{CE},\mathcal{A},b}(k) \;\leq\; Q_R \cdot \mathbf{Adv}^{\text{enc-ind-atk}}_{\mathsf{AE},\mathcal{B}_{\mathsf{AE}},b}(k) + 2Q_R \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{B}_{\mathsf{DS}}}(k)\;,$$

where $\mathcal{B}_{\mathsf{DS}}$ and $\mathcal{B}_{\mathsf{AE}}$ both run in time $\text{Time}(\mathcal{A}) + c \log Q_R$ for a small constant c, and the advantages describe the success probabilities of $\mathcal{B}_{\mathsf{DS}}$ and $\mathcal{B}_{\mathsf{AE}}$ attacking the underlying encryption and signature scheme, respectively (see Appendix A for formal definitions).

The proof idea is as follows. We turn a successful adversary $\mathcal{A}$ on the certified encryption scheme into an adversary $\mathcal{B}_{\mathsf{AE}}$ on the underlying encryption scheme. This algorithm $\mathcal{B}_{\mathsf{AE}}$ tries to guess in advance which of the registered keys adversary $\mathcal{A}$ will use to break the security of the certified scheme. This simulation works as long as adversary $\mathcal{A}$ does not use an unregistered but valid key, in which case we derive a successful attack on the signature scheme used in the certification procedure.

Efficient Certified Encryption Scheme. In the sequel we present our ElGamal-based encryption scheme with implicit certificates. We show that if the computational Diffie-Hellman problem is hard (see Appendix C for a precise statement of this assumption), our scheme guarantees IND-CCA security. At the same time, the efficiency of our scheme is close to that of the basic ElGamal encryption *without* certificate verifications. The security of the following construction is captured by Theorem 3.5. Its security is also provided in Appendix 3.5.

The idea of our scheme is to let the CA issue certificates in forms of Schnorr signatures for identity ID and to use these values for a CCA2-version of the ElGamal encryption. That is, for the CA's public key $pk_{\mathsf{CA}} = Z = g^z$ the CA hands the user the values $R = g^r$ and $\log_g RZ^c$ for $c = H(R, \mathsf{ID})$. To send the user encrypted messages one uses the value RZ^c as the public ElGamal key, and the user can decrypt with his decryption key $sk = \log_g RZ^c$. Below we use a slightly different variant in which the user contributes to the Schnorr signature via a random value $S = g^s$, in order to deny the CA knowledge of the decryption key.

Construction 3.4 ***[Certified ElGamal Encryption]*** *We construct the certified ElGamal encryption scheme* $\mathsf{CE} = (\mathcal{EG}, \mathcal{EK}, (\mathcal{C},\mathcal{U}), \mathcal{E}, \mathcal{D})$ *as follows:*

- Parameter generation: *Algorithm $\mathcal{EG}$ on input 1^k generates a (description of a) group $\mathcal{G}$ of prime order $q = q(k)$, as well as a generator g of this group. Let $2^k \leq q < 2^{k+1}$. Algorithm $\mathcal{EG}$ also picks (descriptions of) hash functions $F = \{0,1\}^* \to \mathbf{Z}_q$, $G : \{0,1\}^* \to \{0,1\}^{t+k}$, $H : \{0,1\}^* \to \mathbf{Z}_q$. It returns $I = (\mathcal{G}, q, g, F, G, H)$. The associated message space is $\{0,1\}^t$. These parameters are given to all parties and algorithms as additional input.*
- Key generation: *Algorithm $\mathcal{EK}$ on input I selects $z \overset{\$}{\leftarrow} \mathbf{Z}_q$ and computes $Z = g^z$. It returns $(pk_{\mathsf{CA}}, sk_{\mathsf{CA}}) = (Z, (Z, z))$.*
- Key registration: *The pair $(\mathcal{C},\mathcal{U})$ of interactive algorithms is defined by the following steps. $\mathcal{C}$ gets as input $sk_{\mathsf{CA}} = (Z, z)$, while $\mathcal{U}$ gets some identity ID and $pk_{\mathsf{CA}} = Z$. The authority $\mathcal{C}$ first picks $r \overset{\$}{\leftarrow} \mathbf{Z}_q$, computes $R = g^r$ and and sends R to $\mathcal{U}$. User $\mathcal{U}$ chooses $s \overset{\$}{\leftarrow} \mathbf{Z}_q$, computes $S = g^s$ and sends (S, ID) back to $\mathcal{C}$. Upon receiving (S, ID) algorithm $\mathcal{C}$ sets $c = H(R, S, \mathsf{ID})$ and $y = r + cz \bmod q$. Let $pk = (R, S)$ and $\mathsf{cert} = \varepsilon$ be empty. $\mathcal{C}$ returns (R, y) to $\mathcal{U}$ and outputs $(\mathsf{ID}, pk, \mathsf{cert})$. $\mathcal{U}$ verifies that $g^y = RZ^c$ for $c = H(R, S, \mathsf{ID})$, computes $sk = s + y \bmod q$ and outputs $(\mathsf{ID}, pk, sk, \mathsf{cert})$. Note that $sk = \log_g RSZ^c$.*
- Encryption: *For input $\mathsf{ID}, pk = (R, S), \mathsf{cert} = \varepsilon, pk_{\mathsf{CA}} = Z$ and message $M \in \{0,1\}^t$ the encryption algorithm picks $\alpha \overset{\$}{\leftarrow} \{0,1\}^k$, computes $a = F(\mathsf{ID}, pk, \mathsf{cert}, \alpha||M)$, $A = g^a$ and $B = G(\mathsf{ID}, pk, \mathsf{cert}, (RSZ^c)^a) \oplus \alpha||M$ where $c = H(R, S, \mathsf{ID})$. It outputs $C = (A, B)$.*
- Decryption: *For input $\mathsf{ID}, sk, \mathsf{cert} = \varepsilon, pk_{\mathsf{CA}} = Z$ and $C = (A, B)$ the decryption algorithm computes $\alpha||M = B \oplus G(\mathsf{ID}, pk, \mathsf{cert}, A^{sk})$ and verifies that $A = g^{F(\mathsf{ID}, pk, \mathsf{cert}, \alpha||M)}$. In this case it returns M, else it returns $\perp$.* ▌

Theorem 3.5 *Suppose that the parameter generator $\mathcal{EG}$ in the encryption scheme in Construction 3.4 generates CDH-secure groups, and that F, G, H are modeled as random oracles. Then the scheme CE in Construction 3.4 is IND-CCA secure in the random oracle model.* ▌

The efficiency of our scheme is comparable to the one of regular ElGamal encryption *without* certificate verification. With the square-and-multiply exponentiation method, basic ElGamal encryption without certification needs $3k$ expected multiplications, our scheme based on implicit certificates requires $3.25k$ multiplications on the average, whereas regular ElGamal encryption with explicit Schnorr signature certificates would require $4.75k$ expected modular multiplications.

4 Secure Signatures in Public-Key Infrastructures

Syntax of Certified-Signature Schemes. A *certified-signature scheme* is a tuple $\mathsf{CS} = (\mathcal{SG}, \mathcal{K}, (\mathcal{C},\mathcal{U}), \mathcal{S}, \mathcal{V})$, where the constituent algorithms run in polynomial time and are defined as follows.

- Algorithms $\mathcal{SG}, \mathcal{K}$ and registration protocol $(\mathcal{C},\mathcal{U})$ are as in the definition of certified encryption schemes (here, $\mathcal{SG}$ replaces $\mathcal{EG}$).
- $\mathcal{S}$ is a (possibly) randomized *signing* algorithm. It takes input an identity ID, a secret key sk, a certificate cert, the authority's public key pk_{CA} and a message $M \in \{0,1\}^*$, and outputs a signature σ.
- $\mathcal{V}$ is a deterministic *verification* algorithm. It takes input an identity ID, a public key pk, a certificate cert, a public key pk_{CA}, a message M and a signature σ, and outputs 0 or 1. In the latter case, we say that σ is a *valid* signature for M relative to $(\mathsf{ID}, pk, \mathsf{cert}, pk_{\mathsf{CA}})$.

We require that for all $M \in \{0,1\}^*$ and all users ID, if (pk, sk) is a key pair for user ID with cert, i.e., $((\mathsf{ID}, pk, \mathsf{cert}), (\mathsf{ID}, pk, sk, \mathsf{cert})) \overset{\$}{\leftarrow} (\mathcal{C}(sk_{\mathsf{CA}}), \mathcal{U}(\mathsf{ID}, pk_{\mathsf{CA}}))$ for $(pk_{\mathsf{CA}}, sk_{\mathsf{CA}})$ generated by $\mathcal{K}(I)$ and I output by $\mathcal{G}(1^k)$, then, for verification, $\mathcal{V}(\mathsf{ID}, pk, \mathsf{cert}, pk_{\mathsf{CA}}, M, \mathcal{S}(\mathsf{ID}, sk, \mathsf{cert}, pk_{\mathsf{CA}}, M)) = 1$.

Security of Certified-Signature Schemes. We consider again a powerful adversary whose capabilities combine the more standard chosen-message attacks with additional capabilities specific to our setting. The adversary attempts a forgery by outputting a user identity, a public key, a message and a signature. Roughly, the adversary wins if the signature is valid with respect to the chosen public key, and either (1) the honest user has registered the public key and has not priorly signed the message, (2) the public key has not been registered, or (3) the same public key has been registered by a different (honest) user. Condition (1) corresponds to the notion of existential unforgeability [22] for standard digital signature schemes, and we require that it holds even if the CA is corrupt. Condition (2) guarantees that signatures for keys that are not bound to identities of the users (i.e., "outside of the PKI") are not valid. Condition (3) prevents attacks where for example a malicious user claims authorship of a message signed by another user.

Definition 4.1 [Security of Certified-Signature Schemes] *Let $\mathsf{CS} = (\mathcal{SG}, \mathcal{K}, (\mathcal{C},\mathcal{U}), \mathcal{S}, \mathcal{V})$ be a certified-signature scheme. We associate to scheme CS, an adversary $\mathcal{A}$, and security parameter k an experiment $\mathbf{Exp}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k)$. The experiment maintains arrays RegListPub, RegListSec which are as in the experiments defining security for certified encryption schemes. In the beginning of the experiment, public parameters are generated via $I \overset{\$}{\leftarrow} \mathcal{G}(1^k)$ and are given as input to the adversary, and then, $\mathcal{A}$ can make the following requests or queries:*

- Corruption of certification authority: *This stage is as in the experiment defining the security of certified encryption.*
- Registering keys of users: *This is handled as in the model for defining security of certified encryption.*
- Signature queries: *$\mathcal{A}$ can make signature requests to a universal signing oracle $\mathcal{US}^{pk_{\mathsf{CA}}}_{\mathsf{CS}}$: on a query $(\mathsf{ID}, pk, \mathsf{cert}, M)$ the oracle verifies that user ID is honest, and if so it looks up the corresponding entry $(\mathsf{ID}, pk, sk, \mathsf{cert})$ in RegListSec and returns to $\mathcal{A}$ a signature $\mathcal{S}(\mathsf{ID}, sk, \mathsf{cert}, pk_{\mathsf{CA}}, M)$. Otherwise, the answer of the oracle is $\perp$.*

Eventually, $\mathcal{A}$ stops and outputs an attempted forgery $(\mathsf{ID}, pk, \mathsf{cert}, M, \sigma)$. The experiment returns 1 if $\mathcal{V}(pk_{\mathsf{CA}}, \mathsf{ID}, pk, \mathsf{cert}, M, \sigma) = 1$ and the following conditions are satisfied (otherwise it returns 0):

1. *ID is honest, and no valid signing query $(\mathsf{ID}, pk, \mathsf{cert}', M)$ was made for any cert', or*
2. *CA is honest and $(\mathsf{ID}, pk, \mathsf{cert}') \notin$ RegListPub, for any cert' (i.e. the user ID never registered the key pk), or*

3. CA *is honest and* $(\mathsf{ID}', pk, \mathsf{cert}') \in RegListPub$ *for some honest user* $\mathsf{ID}' \neq \mathsf{ID}$ *(i.e. some honest user registered pk),*

We define the advantage of adversary $\mathcal{A}$ *as*

$$\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k) = 1\,\right].$$

We say that CS *is a secure certified-signature scheme if the function* $\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(\cdot)$ *is negligible for all poly(k)-time adversaries* $\mathcal{A}$. ▌

"Traditional" certified signature schemes. Here we analyze the traditional approach to certified signatures, where the public-keys of users are certified by the certification authority using a a digital signature scheme. In turn, users produce signatures by using the secret keys associated with their certificated public-keys. Signature verification consist in verifying the signature of the user and the validity of the certificates for the users' public-keys. An interesting aspect that we clarify is that proofs of knowledge of the secret key associated to the public key of the user are not necessary to ensure security of the scheme. We show that simply signing a designated message in a proof of possession is sufficient for security. We now give the scheme (Construction 4.2) and state our security result (Theorem 4.3). The proof is inAppendix **??**, along with the concrete security result.

Construction 4.2 (Traditional Certified Signature Scheme) *Let* $\mathsf{DS} = (\mathcal{SG}, \mathcal{SK}, \mathcal{S}_1, \mathcal{V}_1)$ *be a digital signature scheme*[4]. *The first two algorithms of a certified-signature scheme* $\mathsf{TCS} = (\mathcal{G}, \mathcal{K}, (\mathcal{C}, \mathcal{U}), \mathcal{S}, \mathcal{V})$ *are those of* DS, *and the rest of polynomial time algorithms are defined as follows.*

- Parameter and key generation: $\mathcal{G} \equiv \mathcal{SG}, \mathcal{K} \equiv \mathcal{SK}$.
- Registration: *To register pk, a user* ID *sends pk to the* CA. CA *sends to the user a random "challenge" message*[5] $M' \xleftarrow{\$} \{0,1\}^k$. *The user computes* $\sigma' \xleftarrow{\$} \mathcal{S}(sk, 0||M')$ *and sends it to* CA. *If* $\mathcal{V}(pk, 0||M', \sigma') = 1$ *then* CA *computes* $\mathsf{cert} \xleftarrow{\$} \mathcal{S}(sk_{\mathsf{CA}}, (\mathsf{ID}, pk))$, *sends* cert *to the user and outputs* $(\mathsf{ID}, pk, \mathsf{cert})$. *The user outputs* $(\mathsf{ID}, pk, sk, \mathsf{cert})$.
- Signing: $\mathcal{S}$ *on input* $(\mathsf{ID}, sk, \mathsf{cert}, pk_{\mathsf{CA}}, M)$ *outputs* $\sigma \xleftarrow{\$} \mathcal{S}_1(sk, 1||M)$.
- Verification: $\mathcal{V}$ *takes* $(\mathsf{ID}, pk, \mathsf{cert}, pk_{\mathsf{CA}}, M, \sigma)$. *It outputs* 1 *iff* $\mathcal{V}_1(pk_{\mathsf{CA}}, (\mathsf{ID}, pk), \mathsf{cert}) = 1$ *and* $\mathcal{V}_1(pk, 1||M, \sigma) = 1$. ▌

Theorem 4.3 *Let* $\mathsf{DS} = (\mathcal{SG}, \mathcal{SK}, \mathcal{S}, \mathcal{V})$ *be a digital signature scheme. Then if* DS *is secure (existentially unforgeable under chosen-message attack), then* TCS *is a secure certified signature scheme.* ▌

More precisely, for adversary $\mathcal{A}$ let Q_R be the maximal number of (initiated) registration executions during experiment $\mathbf{Exp}^{\text{cs-uf}}_{\mathsf{TCS},\mathcal{A}}(k)$. Then there exists an algorithm $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{TCS},\mathcal{A}}(k) \;\leq\; 3Q_R \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{B}}(k) + \frac{Q_R}{2^k}.$$

where $\mathcal{B}$ runs in time $\text{Time}(\mathcal{A}) + c \log Q_R$ for a small constant c.

The proof idea is to transform an attacker against the certified signature scheme into one against the underlying signature scheme (by guessing the right target key in advance). It is not hard to see that each successful attack on the certified scheme (new signatures under keys of honest users, generating an unregistered but valid key, and registering keys of honest users under different names) immediately yields a forgery for the signature scheme.

Efficient certified signature schemes. Here we give a construction of an efficient, provably secure certified signature scheme based on Schnorr signatures. Its security, captured by Theorem 4.5, is based on the discrete logarithm assumption (a precise definition is given in Appendix E). The idea is similar to the encryption case, where the CA issued Schnorr signatures to be used as the secret and public ElGamal keys by users, only this time we let the users deploy the key pairs for Schnorr signatures themselves.

[4]For simplicity we consider a case when the certification authority and a user use a single signature scheme. The definition and other results can be easily modified to accommodate a case when different signatures are used by the parties.

[5]We need that all challenge messages be different with overwhelming probability. An alternative approach would be to include a current date and time in the challenge message.

Construction 4.4 (Schnorr-based Certified Signature Scheme) *We define scheme* $\mathsf{CS} = (\mathcal{SG}, \mathcal{K}, (\mathcal{C}, \mathcal{U}), \mathcal{S}, \mathcal{V})$ *by the algorithms:*

- Parameter generation: *Algorithm* $\mathcal{SG}$ *on input* 1^k *generates a (description of a) group* $\mathcal{G}$ *of prime order* $q = q(k)$*, as well as a generator* g *of this group. Let* $2^k \le q < 2^{k+1}$*. Algorithm* $\mathcal{SG}$ *also picks (descriptions of) hash functions* $G : \{0,1\}^* \to \mathbf{Z}_q$*,* $H : \{0,1\}^* \to \mathbf{Z}_q$*. It returns* $I = (\mathcal{G}, q, g, G, H)$*. These parameters are given to all parties and algorithms as additional input.*
- Key generation: *Algorithm* $\mathcal{EK}$ *on input* I *selects* $z \overset{\$}{\leftarrow} \mathbf{Z}_q$ *and computes* $Z = g^z$*. It returns* $(pk_{\mathsf{CA}}, sk_{\mathsf{CA}}) = (Z, (Z, z))$*.*
- Key registration: *The pair* $(\mathcal{C}, \mathcal{U})$ *of interactive algorithms is defined by the following steps.* $\mathcal{C}$ *gets as input* $sk_{\mathsf{CA}} = (Z, z)$*, while* $\mathcal{U}$ *gets some identity* ID *and* $pk_{\mathsf{CA}} = Z$*. The authority* $\mathcal{C}$ *first picks* $r \overset{\$}{\leftarrow} \mathbf{Z}_q$*, computes* $R = g^r$ *and sends* R *to* $\mathcal{U}$*. User* $\mathcal{U}$ *chooses* $s \overset{\$}{\leftarrow} \mathbf{Z}_q$*, computes* $S = g^s$ *and sends* (S, ID) *back to* $\mathcal{C}$*. Upon receiving* (S, ID) *algorithm* $\mathcal{C}$ *sets* $c = H(R, S, \mathsf{ID})$ *and* $y = r + cz \bmod q$*. Let* $pk = (R, S)$ *and* $\mathsf{cert} = \varepsilon$*.* $\mathcal{C}$ *returns* (R, y) *to* $\mathcal{U}$ *and outputs* $(\mathsf{ID}, pk, \mathsf{cert})$*.* $\mathcal{U}$ *verifies that* $g^y = RZ^c$ *for* $c = H(R, S, \mathsf{ID})$*, computes* $sk = s + y \bmod q$ *and outputs* $(\mathsf{ID}, pk, sk, \mathsf{cert})$*. Note that* $sk = \log_g RSZ^c$*.*
- Signing: *For input* ID*,* sk*,* $\mathsf{cert} = \varepsilon$*,* (R, S)*, certificate* ε *and* Z *the signing algorithm picks* $a \overset{\$}{\leftarrow} \mathbf{Z}_q$ *and computes* $A = g^a$ *and* $B = a + sk \cdot G(\mathsf{ID}, A, M)$*. The signature is* $\sigma = (A, B)$*.*
- Verification: *For input* ID*,* $pk = (R, S)$*,* ε*,* $pk_{\mathsf{CA}} = Z$*, a message* M *and a signature* $\sigma = (A, B)$ *the verification algorithm outputs* 1 *if the equation* $g^B = A(RSZ^c)^d$ *holds, where* $c = H(R, S, \mathsf{ID})$ *and* $d = G(\mathsf{ID}, A, M)$*. Otherwise it outputs* 0*.* ▌

Theorem 4.5 *The certified-signature scheme of Construction 4.4 is secure in the random oracle model if the parameter generation algorithm generates DL-secure groups.* ▌

For the above scheme, signing is exactly as in standard Schnorr signature schemes and thus as efficient. Verification of a signature, however, now requires on the average only $1.875k$ modular multiplications with the square-and-multiply method, as opposed to $3.5k$ modular multiplications as required to verify two separate Schnorr signatures.

5 Extensions

5.1 Static vs. Adaptive Corruptions

For simplicity, we allow only static corruptions of the CA and the users in our models. As we discuss next, allowing adaptive corruptions of the parties (except for adaptive user corruptions for encryption) does not give the adversary much advantage over static corruptions.

We first show that adaptive CA corruptions, where the adversary may decide upon corrupting the CA at any point during the execution and, if so, then learns the secret key sk_{CA} as well as all internal randomness of the CA, does not enable much more powerful attacks, neither for signatures nor for encryption. That is, we can guess in advance with probability $1/2$ whether the adaptive adversary will corrupt the CA during the experiment or not. Depending on this prediction we possibly corrupt the CA at the beginning. Then we let the (corrupt) CA honestly follow its program and if the adversary later decides to corrupt the CA, then we hand over the secret key and the randomness. If our initial guess is wrong we stop with fixed output 0. Then we have reduced static CA corruptions to adaptive ones, and this decreases the advantage only by a factor of $1/2$ for both encryption and signatures.

As for adaptive user corruptions, in case of encryption users cannot be corrupted adaptively once they have been activated for the first time. This prevents trivial attacks in which the adversary asks the encryption oracle to create a ciphertext for an honest user and registered key, and then corrupts this user adaptively to learn which message has been encrypted and the bit b. Security against such attacks would require sophisticated solutions like non-committing encryption [] and is beyond the scope of this paper.

For signatures, users may be corrupted adaptively on the other hand (and this can be combined with adaptive corruptions of the CA). Analogously to the case of CA corruptions it follows that such adaptive user corruptions do not increase the adversary's power significantly. Namely, if there are at most n active users during an attack then we can guess in advance with probability $1/(n+1)$ a single user which remains honest (or if no users stays honest). Then we corrupt all other users and let them follow the prescribed program up to the point where the adaptive adversary asks to corrupt one of them. In this case, we give the adversary the secret keys and the randomness of the corresponding user. If we later find out that our guess has been wrong we stop with output 0. Else, the advantage of the adaptive adversary reduces (by a factor $n+1$) to the advantage of breaking the scheme with static user corruptions.

5.2 Hierarchical Certification

Another simplification of our model is that we have a flat certification hierarchy, i.e., each user key is only certified by one CA. In reality, the certification process is usually distributed over several CA's, typically ordered hierarchically in a tree. In such a tree, starting with the root CA, any CA certifies the public keys of its children down to the leave CAs which then certify the user's keys. Our models can be easily extended to this case, as explained next.

We assume that there is a set of CA identities $\mathsf{CA} \in \{0,1\}^*$ with one distinguished root CA identity $\mathsf{rCA} \in \{0,1\}^*$. We further suppose that the set of CA identities is disjoint from the set of user identities $\mathsf{ID} \in \{0,1\}^*$, i.e., the CA keys can only be used for certification tasks. At the beginning of an attack the adversary decides upon corruption of the root CA and a (possibly adversarial) root key pk_{rCA} is published. During the attack the adversary may now also initiate CA registrations between different authorities $\mathsf{CA}, \mathsf{CA}^*$, where the common input is a certification path $(\mathsf{CA}_0, pk_0, \mathsf{cert}_0), (\mathsf{CA}_1, pk_1, \mathsf{cert}_1), \ldots, (\mathsf{CA}_m, pk_m, \mathsf{cert}_m)$ of CA names and keys, starting with the root authority $\mathsf{CA}_0 = \mathsf{rCA}$, $pk_0 = pk_{\mathsf{rCA}}$, $\mathsf{cert}_0 = \epsilon$ and ending with authority $\mathsf{CA}_m = \mathsf{CA}$. At the end of this execution CA outputs $(\mathsf{CA}^*, pk^*, \mathsf{cert}^*)$ and CA^* outputs $(\mathsf{CA}^*, pk^*, sk^*, \mathsf{cert}^*)$ (a party can also output $\perp$ in case of error). This implies in a straightforward way a list of registered public (and secret) keys *RegListPub* (and *RegListSec*) indexed by the certification path.

In the case of a flat hierarchy and a single CA the adversary may put any encryption request where the user is honest and the key registered with the CA, or where the CA is still honest but the public key of the user not registered. In the hierarchical setting with root authority rCA this transfers accordingly. Each encryption query now consists of a certification path $(\mathsf{CA}_0, pk_0, \mathsf{cert}_0), (\mathsf{CA}_1, pk_1, \mathsf{cert}_1), \ldots, (\mathsf{CA}_m, pk_m, \mathsf{cert}_m)$ of CA names and keys (starting again with the root $\mathsf{CA}_0 = \mathsf{rCA}$), as well as a user key $(\mathsf{ID}, pk, \mathsf{cert})$ and two messages M_0, M_1. The requirement now is that user ID is still honest and $(\mathsf{ID}, pk, \mathsf{cert})$ has been registered with authority CA_m, i.e., appears in the array *RegListPub* indexed by the certification path. Or, there must exist an index $i \in \{0,1,2,\ldots,m\}$ such that authorities $\mathsf{CA}_0, \ldots, \mathsf{CA}_i$ are honest but $(\mathsf{CA}_{i+1}, pk_{i+1}, \mathsf{cert}_{i+1})$ (or $(\mathsf{ID}, pk, \mathsf{cert})$ in case $i = m$) is not listed with authority CA_i in the array indexed by the corresponding prefix of the certification path at this point.

Decryption queries are now dealt with accordingly. They also take a certification path as input, in addition to $(\mathsf{ID}, pk, \mathsf{cert}, C)$. If C has never been returned by encryption oracle for a query including $(\mathsf{ID}, pk, \mathsf{cert})$, then the decryption oracle looks for an entry in the array *RegListSec* indexed by the certification path. If such an entry exists then the oracle decrypts the ciphertext, otherwise it returns $\perp$.

For signature schemes and oracle queries $(\mathsf{ID}, pk, \mathsf{cert}, M)$, including a certification path, the signature oracle verifies that the user ID is still honest and that $(\mathsf{ID}, pk, \mathsf{cert})$ appears in the array index by the path. If so, it looks up the corresponding secret key and signs, else it returns $\perp$. The verification algorithm also gets

Our definition of a certified signature scheme immediately allows to build such hierarchical certified schemes. Namely. suppose one uses a certified signature scheme according to Definition 4.1 to let CAs issue certificates for keys, and that the validity of each key in the certification path when encrypting or verifying signatures is checked. Then it follows almost identical to the proofs of Theorems 3.3 and 4.3 that this yields a secure certified encryption or signature scheme in the hierarchical setting.

5.3 Integrating Revocation

Augmenting our security definitions for encryption and signatures by explicit revocation in forms of *certificate revocation lists* (CRLs) or through the *online certificate status protocol* (OCSP) is straightforward. Namely, let the adversary also allow to ask for revocation of certificates during the experiment. For such a query the key $(\mathsf{ID}, pk, \mathsf{cert})$ is marked as revoked and, from this point on, may not be used in subsequent encryption or signature queries. Note that this only limits the adversary's power.

Implicit revocation can be added to our definitions by considering extended identities $\mathsf{ID} = \mathsf{ID}_0||\langle i \rangle$ with augmented time periods i and by attaching round scheduling in a straightforward way. A user can then refresh his or her certificate for the next time period $i+1$ by simply running the registration protocol with the CA and identity $\mathsf{ID} = \mathsf{ID}_0||\langle i+1 \rangle$. The CA can keep state such that it can differentiate between new registrations and refreshes. By this, we easily capture implicit revocation methods like Micali's hash chains or identity-based schemes. The specifications for successful attacks in both cases remain unchanged, although further restrictions can be incorporated, e.g., that the adversary must corrupt users with identities $\mathsf{ID}_0||\langle j \rangle$ if it already controls the user $\mathsf{ID}_0||\langle i \rangle$ for a previous period $i < j$, or that only encryption and signature queries for the current time period are allowed.

Modifying our schemes to work with Micali's hash chains, for example, is straightforward.

6 Related Work

Here we review several PKI-related works in the literature and put our results in the context [20, 2, 14, 15, 27, 28, 40, 26, 33, 29, 39, 16, 7, 17, 41, 42, 24, 30].

Gentry [20], Al-Riyami and Paterson [2] and subsequent works [29, 39, 16, 7, 17, 3, 41, 42, 24, 30] recently proposed public-key encryption schemes that do not assume a standard PKI. Similarly to our efficient scheme, certificates are implicit, that is, a sender does not have to verify the certificate before sending an encrypted message, yet only the user who properly registered its public key is able to decrypt. The goals of their schemes and ours, however, differ. The motivation for the works of [20, 2, 29, 39, 16, 7, 17, 41, 42, 24, 30] is to overcome the main weakness of identity-based encryption (IBE) [38, 11], namely, the requirement that the trusted party called a private key generator (PKG) knows the secret keys of the users, while preserving the advantage of IBE of simplified management of expired and revoked public keys. Gentry also eliminates the requirement of a secure channel between a user and the PKG. On the other hand, the goal of our scheme is to achieve an efficiency improvement over "traditional" discrete-logarithm-based certified encryption schemes and, similar, for certified signature schemes. While our schemes require the key management support of a traditional PKI, they come with computational savings.

Security of implicit certificates in the context of digital signatures had been priorly investigated by Brown, Gallant, and Vanstone [12]. Their security model is only concerned with the certification process and does not consider the usage of the resulting keys. However, for the particular application analyzed in the paper, the resulting security model (which is strictly weaker and less general than the one we introduce here) appears to suffice.

Canetti [14] recently presented a universally composable certification protocol, that uses traditional signature-based certificates. And while universal composition provides very strong security guarantees, his approach falls again short of investigating the combined certification and encryption or signature process, neither does his model take CA corruptions and the broader adversarial capabilities into account. In contrast to our more efficient solutions based on implicit certificates, alternatives to the traditional approach are not discussed in [14].

Stronger security requirements on signatures are imposed by the model suggested by Menezes and Smart [34] for the use of signatures in the "multi-user setting". Condition (3) of our definition of security for signature schemes is reminiscent of their security requirement. However, the framework proposed in [34] does not explicitly consider the registration protocol.

Since our basic model is concerned with security under one-level of certification, some of the issues specific to hierarchical PKIs do not show up explicitly. When tackling such settings (which naturally arise in practice, e.g. when using PGP [43]), special attention needs to be payed to the trust that parties put in certificate chains, given that one or several of the CAs could have been corrupted. Prior research that could prove useful in extending our framework to these settings include various models for trust in key authenticity [32, 13, 25], quantitative trust evaluation [8], as well as various defenses against multiple CA corruption (e.g. multi-certificate chains [36]).

Our efficient certified encryption scheme resembles the PKI-enabled CA-Oblivious encryption scheme independently proposed by Castelluccia et al. in their recent work [15] as a building block for a secret handshake protocol. The authors analyze their schemes with respect to a significantly weaker security notion, namely one-wayness. Moreover, in their scheme the CA knows the secret keys of the users and is trusted to behave honestly. In their model, it is also assumed that the CA is trusted not to use the knowledge of the secret keys. Moreover, no standard outsider attacks are considered, that is a scheme where an adversary can decrypt messages addressed to the registered users can be proven secure! The authors prove that their scheme satisfies their security notion in the RO model. The authors suggest how to modify the scheme to allow the CA to be less trusted, but the security of the resulting scheme is unclear. It is also suggested in [15] that the scheme can be made IND-CCA secure using the Fujisaki-Okamoto transform [19]. We note, however, that it is not immediate without a new proof that this would work, since the Fujisaki-Okamoto transform can be provably applied to basic encryption schemes that assume a standard PKI with explicit certificates. The primitive of [15] is different; it employs implicit certificates. Therefore, a new security definition and a proof will be necessary to validate this suggestion. On the other hand, our scheme provably satisfies a very strong security definition discussed above, where the CA is only trusted not to register new keys for users without their permission. Our results also show that the Fujisaki-Okamoto transform is not needed as our scheme is very simple and yet IND-CCA secure.

Our efficient certified signature scheme resembles the proxy signature scheme of Kim, Park and Won [27] and the self-certified signature scheme of Lee and Kim [28]. Unlike our scheme, the proxy signature scheme assumes a PKI where each user already holds a public key and a digital certificate. Neither of these papers provides formal security definitions and analyses. A modification of the proxy signature scheme of [27] has been proven secure in [10], but their proof is for a primitive that differs from ours in that it assumes a PKI, explicit certificates, and involves a different security notion.

7 Acknowledgments

Part of the work was done while the authors were at UCSD. We thank Martin Abadi, Mihir Bellare and Burt Kaliski for useful discussions, and the anonymous reviewers for their comments. Alexandra Boldyreva is supported in part by NSF CAREER award 0545659. Marc Fischlin is supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). Bogdan Warinschi is supported in part by the ACI Jeunes Chercheurs JC 9005.

References

[1] C. Adams and S. Farrell. Internet x.509 public key infrastructure: Certificate management protocols. Work in progress, 2004.

[2] S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, volume 2894 of *LNCS*, pages 452–473. Springer-Verlag, 2003.

[3] J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless public key encryption without pairing. In *Information Security (ISC)*, volume 3650 of *LNCS*, pages 134–148. Springer-Verlag, 2005.

[4] M. Bellare, A. Boldyreva, and J. Staddon. Multi-recipient encryption schemes: Security notions and randomness re-use. In *Public-Key Cryptography (PKC)*, volume 2567. Springer-Verlag, 2003.

[5] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*. IEEE Computer Society, 1997.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Conference on Computer and Communications Security (CCS)*. ACM, 1993.

[7] K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless kems. *Cryptology ePrint Archive, Report 2005/058.*, 2005.

[8] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Computer Security—ESORICS 94*, pages 3–18. Springer-Verlag, 1994.

[9] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In *Public Key Cryptography (PKC) '03*, pages 31–46, 2003.

[10] A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signaure schemes for delegation of signing rights. *Cryptology ePrint Archive, Report 2003/096.*, 2003.

[11] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, volume 2139 of *LNCS*. Springer-Verlag, 2001.

[12] D.R. L. Brown, R. P. Gallant, and S. A. Vanstone. Provably secure implicit certificate schemes. In *Conference on Financial Cryptography '01*, pages 156–165. Springer-Verlag, 2002.

[13] M. Burmester, Y. Desmedt, and G. Kabatianskii. Trust and security: A new look at the Byzantine generals problem. In R. N. Wright and P. G. Neumann, editors, *Network Threats, DIMACS, Series in Discrete Mathematics and Theoretical Computer Science, 1996, vol. 38*. AMS, 1998.

[14] R. Canetti. Universally composable signature, certification, and authentication. In *CSFW*. IEEE Computer Society, 2004.

[15] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from CA-oblivious encryption. In *ASIACRYPT*, volume 3329 of *LNCS*, pages 293–307. Springer-Verlag, 2004.

[16] Z. Cheng and R. Comley. Efficient certificateless public key encryption. *Cryptology ePrint Archive, Report 2005/012.*, 2005.

[17] A. W. Dent and C. Kudla. On proofs of security for certificateless cryptosystems. *Cryptology ePrint Archive, Report 2005/348.*, 2005.

[18] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, Vol. 31, 1985.

[19] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, volume 1666 of *LNCS*, pages 537–554, 1999.

[20] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT*, volume 2656 of *LNCS*, pages 272–293. Springer-Verlag, 2003.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 1984.

[22] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.

[23] J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In *CRYPTO*, volume 2729. Springer-Verlag, 2003.

[24] B. Hu, D. Wong, Z. Zhang, and X. Deng. Key replacement attack against a generic construction of certificateless signature. In *ACISP*, volume 4058 of *LNCS*, pages 235–246. Springer-Verlag, 2006.

[25] A. Josang. Trust-based decision making for electronic transactions. In *Fourth Nordic Workshop on Secure IT Systems (NORDSEC'99)*, pages 99–105, 1999.

[26] B. Kaliski. An unknown key-share attack on the mqv key agreement protocol. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):275–288, 2001.

[27] S. Kim, S. Park, and D. Won. Proxy signatures, revisited. In *ICICS*. Springer-Verlag, 1997.

[28] B. Lee and K. Kim. Self-certified signatures. In *Indocrypt*, volume 2551 of *LNCS*. Springer-Verlag, 2002.

[29] Y.-R. Lee and H.-S. Lee. An authenticated certificateless public key encryption scheme. *Cryptology ePrint Archive, Report 2004/150.*, 2004.

[30] B. Libert and J.-J. Quisquater. On Constructing Certificateless Cryptosystems from Identity Based Encryption. In *Public Key Cryptography (PKC)*, LNCS. Springer-Verlag, 2006.

[31] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography (SAC)*, pages 184–199. Springer-Verlag, 1999.

[32] U. Maurer. Modeling public-key infrastructure. In *Computer Security—ESORICS 96*, pages 325–350. Springer-Verlag, 1996.

[33] A. Menezes, M. Qu, and S. A. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *Selected Areas in Cryptography (SAC)*, 1995.

[34] A. Menezes and N. Smart. Security of signature schemes in a multi-user setting. *Designs, Codes and Cryptography*, 33:261–274, 2004.

[35] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *Conference on Computer and Communications Security (CCS)*, pages 245–254. ACM, 2001.

[36] M. K. Reiter and S. G. Stubblebine. Path independence for authentication in large scale systems. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 57–66, 1997.

[37] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[38] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*. Springer-Verlag, 1984.

[39] Y. Shi and J. Li. Provable efficient certificateless public key encryption. *Cryptology ePrint Archive, Report 2005/287.*, 2005.

[40] V. Shoup. On formal models for secure key exchange. *IBM Research Report RZ 3120*, 1999.

[41] D. H. Yum and P. J. Lee. Generic construction of certificateless encryption. In *EuroPKI*, volume 3043 of *LNCS*, pages 802–811. Springer-Verlag, 2004.

[42] D. H. Yum and P. J. Lee. Generic construction of certificateless signature. In *ACISP*, volume 3108 of *LNCS*, pages 200–211. Springer-Verlag, 2004.

[43] P. R. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, Massachussets, 1995.

Appendix

A Standard definitions

A.1 Asymmetric Encryption Schemes and their Security

Syntax. An asymmetric encryption scheme $\mathsf{AE} = (\mathcal{EG}, \mathcal{EK}, \mathcal{E}, \mathcal{D})$ is specified by four polynomial-time algorithms with the following functionalities. The randomized *parameter-generation* algorithm $\mathcal{EG}$ takes input 1^k, where k is the security parameter, and outputs some global parameters I. These may contain, for example, another security parameter, the description of a cyclic group and a generator, and the description of a hash function. We assume that these parameters become publicly available. The randomized *key-generation* algorithm $\mathcal{EK}$ takes input global parameters I and outputs a pair (pk, sk) consisting of a public key and a matching secret key, respectively. The randomized *encryption* algorithm $\mathcal{E}$ takes input a public key pk and a message M, and outputs a ciphertext C. The deterministic *decryption* algorithm $\mathcal{D}$ takes input a secret key sk and a ciphertext C, and outputs a message M or a special symbol $\perp$ to indicate that the ciphertext is invalid. Associated to I is a *message space* $\mathrm{MsgSp}(I)$ from which M is allowed to be drawn. For any I that can be output by $\mathcal{EG}(1^k)$, any pair of keys (pk, sk) that can be output by $\mathcal{SK}(I)$ and any $M \in \mathrm{MsgSp}(I)$, it is required that $\mathcal{D}(sk, \mathcal{E}(pk, M)) = M$.

Definition A.1 [Security of an asymmetric encryption scheme] *Let* $\mathsf{AE} = (\mathcal{EG}, \mathcal{EK}, \mathcal{E}, \mathcal{D})$ *be an asymmetric encryption scheme. Consider experiments* $\mathbf{Exp}^{\text{enc-ind-cpa}}_{\mathsf{AE},\mathcal{A},b}(k), \mathbf{Exp}^{\text{enc-ind-cca}}_{\mathsf{AE},\mathcal{A},b}(k)$ *associated to* AE*, a bit* $b \in \{0,1\}$ *and an adversary* $\mathcal{A}$*. In both experiments* $\mathcal{A}$ *is given input a public key* I, pk *and access to a left-right encryption oracle* $\mathcal{UE}_{\mathsf{AE}}(pk, b, \cdot, \cdot)$*, where* pk *and* sk *are matching keys generated via* $I \xleftarrow{\$} \mathcal{EG}(1^k)$; $(pk, sk) \xleftarrow{\$} \mathcal{EK}(I)$*. The oracle takes input two messages* $M_0, M_1 \in \mathrm{MsgSp}(I)$ *of equal length and returns a ciphertext* $C \xleftarrow{\$} \mathcal{E}(pk, M_b)$*. In experiment* $\mathbf{Exp}^{\text{enc-ind-cca}}_{\mathsf{AE},\mathcal{A},b}(k)$ *the adversary is also given input a decryption oracle* $\mathcal{D}(sk, \cdot)$*.* $\mathcal{A}$ *queries the oracle(s) on inputs of its choice, with a restriction of not querying its decryption oracle on ciphertexts previously returned by the left-right encryption oracle.* $\mathcal{A}$ *eventually stops and outputs a guess d which is also the output of the experiment. The adversary's advantages in attacking the scheme are defined as follows. For* $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$

$$\mathbf{Adv}^{\text{enc-ind-atk}}_{\mathsf{AE},\mathcal{A}}(k) \quad = \quad \Pr[\mathbf{Exp}^{\text{enc-ind-atk}}_{\mathsf{AE},\mathcal{A},1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{enc-ind-atk}}_{\mathsf{AE},\mathcal{A},0}(k) = 1] \ .$$

AE *is said to be IND-CPA (resp. IND-CCA) secure if the corresponding advantage of any* $\mathrm{poly}(k)$*-time adversary* $\mathcal{A}$ *is negligible in the security parameter* k*.* ▌

We adopt the convention that the *time complexity* of adversary $\mathcal{A}$ is the execution time of the entire experiment, including the time taken for key generation, and computation of answers to oracle queries. The same convention will be used implicitly in other definitions of the paper.

A.2 Digital Signature Schemes and their Security

SYNTAX. A digital signature scheme $\mathsf{DS} = (\mathcal{SG}, \mathcal{SK}, \mathcal{S}, \mathcal{V})$ is specified by four polynomial-time algorithms with the following functionalities. The randomized *parameter-generation* algorithm $\mathcal{SG}$ takes input 1^k, where k is the security parameter, and outputs some global parameters I. The randomized *key-generation* algorithm $\mathcal{SK}$ takes input global parameters I and outputs a pair (pk, sk) consisting of a public key and a matching secret key, respectively. The (possibly) randomized *signing* algorithm $\mathcal{S}$ takes input a secret key sk and a message $M \in \{0,1\}^*$, and outputs a signature σ. The deterministic *verification* algorithm $\mathcal{V}$ takes input a public key pk, a message M and a candidate signature σ for M, and outputs a bit. We say that σ is a *valid* signature for M relative to pk if $\mathcal{V}(pk, M, \sigma) = 1$. For any pair of keys (pk, sk) that can be output by $\mathcal{SK}$ and any $M \in \{0,1\}^*$, it is required that $\mathcal{V}(pk, M, \mathcal{S}(sk, M)) = 1$.

Definition A.2 [Security of a digital signature scheme] *Let* $\mathsf{DS} = (\mathcal{SG}, \mathcal{SK}, \mathcal{S}, \mathcal{V})$ *be a digital signature scheme. Consider an adversary* $\mathcal{A}$ *that is given input a public key pk and access to a signing oracle* $\mathcal{US}_{\mathcal{S}}(sk, \cdot)$*, where pk and sk are matching keys generated via* $I \xleftarrow{\$} \mathcal{SG}(1^k)$; $(pk, sk) \xleftarrow{\$} \mathcal{SK}(I)$*. The oracle takes input a message* M *and returns a signature* $\sigma \xleftarrow{\$} \mathcal{S}(sk, M)$*.* $\mathcal{A}$ *queries this oracle on messages of its choice, and eventually outputs a forgery* (M, σ)*. The adversary's advantage* $\mathbf{Adv}^{\text{ds-uf}}_{\mathsf{DS},\mathcal{A}}(k)$ *in attacking the scheme is the probability that it outputs a pair* (M, σ) *such that* σ *is a valid signature for message* M *and this message was not queried to the signing oracle.* DS *is said to be secure against existential forgery under adaptive chosen-message attacks (or, simply, secure) if the advantage of any* poly(k)*-time adversary* $\mathcal{A}$ *is negligible in the security parameter* k*.* ▌

B Proof of Theorem 3.3

Recall that our goal is to show that the traditional approach of using signature-based certificates provides a secure certified encryption scheme. In terms of concrete security —from which the theorem follows— we show that for every adversary $\mathcal{A}$ there exists adversaries $\mathcal{B}_{\mathsf{DS}}$ and $\mathcal{B}_{\mathsf{AE}}$ such that for atk $\in$ {cpa, cca},

$$\mathbf{Adv}^{\text{cenc-ind-atk}}_{\mathsf{TCE},\mathcal{A},b}(k) \;\le\; Q_R \cdot \mathbf{Adv}^{\text{enc-ind-atk}}_{\mathsf{AE},\mathcal{B}_{\mathsf{AE}},b}(k) + 2Q_R \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{B}_{\mathsf{DS}},\mathsf{DS}}(k) \; ,$$

where Q_R be the maximal number of (initiated) registration executions during experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{TCE},\mathcal{A},0}(k)$ or $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{TCE},\mathcal{A},1}(k)$ and $\mathcal{B}_{\mathsf{DS}}$ and $\mathcal{B}_{\mathsf{AE}}$ both run in time $\text{Time}(\mathcal{A}) + c \log Q_R$ for a small constant c.

Proof:

For the proof define the following attack of algorithm $\mathcal{B}_{\mathsf{AE}}$ on the underlying encryption scheme AE. $\mathcal{B}_{\mathsf{AE}}$ is given I_e and pk as input and is also given access to a left-right encryption oracle for key pk (and possibly a decryption oracle), i.e.,$\mathcal{B}_{\mathsf{AE}}$ runs experiment $\mathbf{Exp}^{\text{enc-ind-cpa}}_{\mathsf{TCE},\mathcal{A},b}(k)$ (or $\mathbf{Exp}^{\text{enc-ind-cca}}_{\mathsf{TCE},\mathcal{A},b}(k)$) for unknown bit b.

Algorithm $\mathcal{B}_{\mathsf{AE}}$ initially picks an integer i between 1 and Q_R. Then it simulates an attack of $\mathcal{A}$ on the certified scheme by providing the same interaction for $\mathcal{A}$ as in experiment $\mathbf{Exp}^{\text{cenc-ind-atk}}_{\mathsf{TCE},\mathcal{A},b}(k)$. The only exceptions lie in the i-th registration step for an honest user and in the way encryption (and decryption) queries are answered. Specifically, if the i-th registration protocol with an honest user ID is initiated then $\mathcal{B}_{\mathsf{AE}}$ sends the given pk on behalf of that user (and possibly obtains a signature cert for ID$||pk$; if so, then (ID, pk, cert) is also listed in *RegListPub*). Call (ID, pk, cert) the *target.* Answer encryption queries $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell, M_{0,\ell}, M_{1,\ell})$ as follows:

- If $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell)$ equals the target then $\mathcal{B}_{\mathsf{AE}}$ passes $(M_{0,\ell}, M_{1,\ell})$ to its encryption oracle and returns the answer to $\mathcal{A}$.
- If $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell)$ is different from the target but does not appear in *RegListPub* (and the CA is honest) then, if cert is a valid signature abort with output 0, else return $\perp$ and continue.
- Else, if $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell)$ is different from the target and appears in *RegListPub* because of the j-th registration with an honest user, $\mathcal{B}_{\mathsf{AE}}$ computes $C \xleftarrow{\$} \mathcal{E}_e(pk_\ell, M_{1,\ell})$ if $j < i$ and $C \xleftarrow{\$} \mathcal{E}_e(pk_\ell, M_{0,\ell})$ if $j > i$. Return C.

If atk = cca then decryption queries are treated as in the original experiment, only that queries involving the target $(\mathsf{ID}, pk, \mathsf{cert})$ are relayed between $\mathcal{A}$ and $\mathcal{B}_{\mathsf{AE}}$'s decryption oracle. If adversary $\mathcal{A}$ finally stops with output g then $\mathcal{B}_{\mathsf{AE}}$ copies this output and stops as well.

Note that $\mathbf{Exp}^{\text{enc-ind-atk}}_{\mathsf{AE},\mathcal{B}_{\mathsf{AE}},b}(k)$ describes the output of the simulation above, given $\mathcal{B}_{\mathsf{AE}}$'s left-right oracle always encrypts message M_b when queried. Define $\mathsf{UnregKey}_b$ to be the event that, during the simulation, the adversary submits a query $(\mathsf{ID}, pk, \mathsf{cert}, M_0, M_1)$ to the universal encryption oracle such that the tuple $(\mathsf{ID}, pk, \mathsf{cert})$ does not appear in *RegListPub* at the time of submission, that the CA is honest and that the oracle returns a ciphertext $C \neq \perp$. A standard hybrid argument, conditioning on events $\mathsf{UnregKey}_0$, $\mathsf{UnregKey}_1$, then shows

$$\mathbf{Adv}^{\text{cenc-ind-atk}}_{\mathsf{AE},\mathcal{B}_{\mathsf{AE}}}(k) \geq \frac{1}{Q_R} \cdot \mathbf{Adv}^{\text{cenc-ind-atk}}_{\mathsf{TCE},\mathcal{A}}(k) - 2 \cdot \max\left\{\Pr\left[\,\mathsf{UnregKey}_0\,\right], \Pr\left[\,\mathsf{UnregKey}_1\,\right]\right\} \; .$$

Furthermore, it can be easily seen that the probability of event $\mathsf{UnregKey}_b$ (for arbitrary but fixed b) is bounded by the success probability of forging signatures:

$$\Pr\left[\,\mathsf{UnregKey}_b\,\right] \leq \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{B}_{\mathsf{DS}},\mathsf{DS}}(k) \; ,$$

for an adversary $\mathcal{B}_{\mathsf{DS}}$ running in time $\mathrm{Time}(\mathcal{A}) + c \log Q_R$. ▌

C CDH-secure Groups and Proof of Theorem 3.5

In this section we prove security of the efficient certified encryption scheme under the DH assumption:

Diffie Hellman Groups. We say that the parameter generator $\mathcal{EG}$ in Construction 3.4 *generates CDH-secure groups* if for any adversary A running in expected polynomial time, the following is negligible in k:

$$\mathbf{Adv}^{DH}_{\mathcal{EG},A}(k) = \Pr\left[\, (\mathcal{G}, q, g, F, G, H) \xleftarrow{\$} \mathcal{EG}(1^k);\; a, b \xleftarrow{\$} \mathbf{Z}_q \;:\; A(\mathcal{G}, q, g, g^a, g^b) = g^{ab} \,\right].$$

Proof of Theorem 3.5. Let $Q_{\mathcal{UE}}$, $Q_{\mathcal{D}}$, Q_F, Q_G, Q_H and Q_R be the maximum number of queries an adversary $\mathcal{A}$ makes in experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},0}(k)$ or $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k)$ to oracles $\mathcal{UE}$ and $\mathcal{UD}$, and F, G and H as well as the maximum number of (initiated) executions of the registration protocol. Then we show that there exist algorithms $\mathcal{DH}_{\text{reg}}$, $\mathcal{DH}_{\text{unreg}}$ such that

$$\begin{aligned}
\mathbf{Adv}^{\text{cenc-ind-cca}}_{\mathsf{CE},\mathcal{A}}(k) \;\leq\; & \frac{2Q_{\mathcal{UE}}(Q_G + Q_{\mathcal{UE}} + Q_{\mathcal{D}})}{2^k} \\
& + \; 2 \cdot \left(Q_{\mathcal{UE}} Q_G \cdot \mathbf{Adv}^{DH}_{\mathcal{EG},\mathcal{DH}_{\text{reg}}}(k) + \frac{(Q_H + Q_R)Q_R}{2^k} + \frac{2(Q_F + Q_{\mathcal{UE}} + Q_{\mathcal{D}})^2}{2^k} + \frac{Q_{\mathcal{D}}}{2^k} \right) \\
& + \; 2 \cdot \left(4Q_G^2 \cdot \mathbf{Adv}^{DH}_{\mathcal{EG},\mathcal{DH}_{\text{unreg}}}(k) + \frac{2(Q_H + Q_R)Q_R + 1}{2^k} + \frac{4(Q_F + Q_{\mathcal{UE}} + Q_{\mathcal{D}})^2}{2^k} + \frac{2Q_{\mathcal{D}}}{2^k} \right)^{1/3} \; ,
\end{aligned}$$

where $\mathcal{DH}_{\text{reg}}$, $\mathcal{DH}_{\text{unreg}}$ run in *expected* time at most $2 \cdot \mathrm{Time}(\mathcal{A}) + 2ctk^3 \cdot (Q_F \log Q_F + Q_G \log Q_G + Q_H \log Q_H)$, for a small constant c. The statement of the theorem follows.

We start with a proof for the simpler case that the adversary does not put any decryption requests and merely mounts a chosen-plaintext attack. We afterwards discuss how to treat decryption queries. Another simplification, which is due to the fact that the adversary does not ask for decryptions, is that the function F now becomes obsolete. We thus first look at the case that the encryption algorithm, instead of computing $a \leftarrow F(\mathsf{ID}, pk, \mathsf{cert}, \alpha || M)$ and encrypting $\alpha || M$, now chooses $a \xleftarrow{\$} \mathbf{Z}_q$ at random and merely encrypts M. For this IND-CPA attack recall that the adversary is given access to a left-right oracle $\mathcal{UE}$ which always encrypts either the message $M_b = M_0$ or the message $M_b = M_1$ with each submission (given that the query is valid), and that this is the only difference between experiments $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},0}(k)$ and $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k)$.

We introduce the following notation for the encryption queries in experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$. Let $\mathsf{ID}_i, pk_i = (R_i, S_i), \mathsf{cert}_i$, $M_{0,i}, M_{1,i}$ be the i-th query of $\mathcal{A}$ to oracle $\mathcal{UE}$. If this query is valid and the oracle chooses a_i and encrypts $M_{b,i}$ as $(g^{a_i}, G(\mathsf{ID}_i, pk_i, \mathsf{cert}_i, T_i) \oplus M_{b,i})$ for $T_i = (R_i S_i Z^{c_i})^{a_i}$ and $c_i = H(\mathsf{ID}_i, R_i, S_i)$, then we call $(\mathsf{ID}_i, pk_i, \mathsf{cert}_i, T_i)$, or sometimes simply T_i, the *i-th target*. We say that the adversary *asks (or queries) about the i-th target* if it submits this target to oracle G during the attack. Let $\mathsf{TargetQuery}_b(i)$ be the event that the adversary first asks about the i-th target during the experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$ before asking about any other target. Let $\mathsf{TargetQuery}_b \equiv \exists i \in \{1, 2, \ldots, Q_{\mathcal{UE}}\} : \mathsf{TargetQuery}_b(i)$. Then,

$$\begin{aligned}\mathbf{Adv}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A}}(k) &= \Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},0}(k) = 1] \\ &\le \Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k) = 1 | \neg\mathsf{TargetQuery}_1] \cdot \Pr[\neg\mathsf{TargetQuery}_1] \\ &\quad - \Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},0}(k) = 1 | \neg\mathsf{TargetQuery}_0] \cdot \Pr[\neg\mathsf{TargetQuery}_0] \\ &\quad + \Pr[\mathsf{TargetQuery}_1] \ .\end{aligned}$$

Given that the adversary never asks about the targets in the experiment, the values $G(\mathsf{ID}_i, pk_i, \mathsf{cert}_i, T_i) \oplus M_{b,i}$ are all random and independent of b. Hence,

$$\Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k) = 1 | \neg\mathsf{TargetQuery}_1] = \Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},0}(k) = 1 | \neg\mathsf{TargetQuery}_0]$$

and

$$\begin{aligned}&\mathbf{Adv}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A}}(k) \\ &\quad \le \Pr[\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},1}(k) = 1 | \neg\mathsf{TargetQuery}_1] \cdot (\Pr[\neg\mathsf{TargetQuery}_1] - \Pr[\neg\mathsf{TargetQuery}_0]) \\ &\qquad + \Pr[\mathsf{TargetQuery}_1] \\ &\quad \le \Pr[\mathsf{TargetQuery}_0] + \Pr[\mathsf{TargetQuery}_1] \le 2 \cdot \max\{\Pr[\mathsf{TargetQuery}_0], \Pr[\mathsf{TargetQuery}_1]\} \ .\end{aligned}$$

Hence, it suffices to bound the probability of event $\mathsf{TargetQuery}_b$ for an arbitrary b. Fix $b \in \{0, 1\}$. For any i and event $\mathsf{TargetQuery}_b(i)$ we distinguish between event $\mathsf{TargetQuery}^{\text{pre}}_b(i)$, that the adversary asks about the i-th target before receiving the answer from $\mathcal{UE}$ for the i-th query, and $\mathsf{TargetQuery}^{\text{post}}_b(i)$, that $\mathcal{A}$ submits T_i to G after having receiving $\mathcal{UE}$'s reply to the i-th query (both events also presume that none of the other targets is asked to G before). Then for

$$\mathsf{TargetQuery}^{\text{pre}}_b \equiv \exists i : \mathsf{TargetQuery}^{\text{pre}}_b(i) \qquad \text{and} \qquad \mathsf{TargetQuery}^{\text{post}}_b \equiv \exists i : \mathsf{TargetQuery}^{\text{pre}}_b(i)$$

we conclude:

$$\begin{aligned}\Pr[\mathsf{TargetQuery}_b] &\le \Pr[\mathsf{TargetQuery}^{\text{pre}}_b] + \Pr[\mathsf{TargetQuery}^{\text{post}}_b] \\ &\le \frac{(Q_G + Q_{\mathcal{UE}})Q_{\mathcal{UE}}}{2^k} + \Pr[\mathsf{TargetQuery}^{\text{post}}_b],\end{aligned}$$

because each of the at most $Q_{\mathcal{UE}}$ targets is a random group element (over the choice of a) and the probability that any of the at most $Q_G + Q_{\mathcal{UE}}$ previous queries to G (including the implicit ones in calls to $Q_{\mathcal{UE}}$) matches this group element is at most $1/q \le 2^{-k}$.

To limit the probability $\Pr[\mathsf{TargetQuery}^{\text{post}}_b]$ we further divide $\mathsf{TargetQuery}^{\text{post}}_b(i)$ according to event $\mathsf{RegKey}_b(i)$ (and event $\mathsf{UnregKey}_b(i)$) that the key $(\mathsf{ID}_i, pk_i, \mathsf{cert}_i)$ in $\mathcal{A}$'s i-th query to $\mathcal{UE}$ appears in the list *RegListPub* of registered keys at this point (or is unregistered and does not appear there). Then,

$$\begin{aligned}\Pr[\mathsf{TargetQuery}^{\text{post}}_b] &\le \Pr[\exists i : \mathsf{TargetQuery}^{\text{post}}_b(i) \wedge \mathsf{RegKey}_b(i)] \\ &\quad + \Pr[\exists i : \mathsf{TargetQuery}^{\text{post}}_b(i) \wedge \mathsf{UnregKey}_b(i)] \ .\end{aligned}$$

We first bound the probability $\Pr[\exists i : \mathsf{TargetQuery}^{\text{post}}_b(i) \wedge \mathsf{RegKey}_b(i)]$. For this we define an algorithm $\mathcal{DH}_{\text{reg}}$ which gets input $(\mathcal{G}, q, g, g^a, g^b)$ and is supposed to output g^{ab}. To do so, $\mathcal{DH}_{\text{reg}}$ runs a simulation of $\mathcal{A}$:

- $\mathcal{DH}_{\text{reg}}$ first picks indices i, j at random between 1 and the number $Q_{\mathcal{UE}}$ of queries to $\mathcal{UE}$, respectively, the number Q_G of queries to oracle G. Algorithm $\mathcal{DH}_{\text{reg}}$ stops (unless it has stopped before) as soon as $\mathcal{A}$ makes the j-th query $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell, X)$ to oracle G. In this case, if $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell)$ appears in list *RegListPub*, then $\mathcal{DH}_{\text{reg}}$ takes the associated values u_i, v_ℓ (as defined below) and outputs $Y = X^{1/(u_i v_\ell)}$.

- In the beginning $\mathcal{DH}_{\text{reg}}$ sets $I = (\mathcal{G}, q, g, G, H)$ where oracles G, H are specified below. $\mathcal{DH}_{\text{reg}}$ starts to simulate an attack of $\mathcal{A}$ in experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$ by impersonating the honest parties, i.e., $\mathcal{DH}_{\text{reg}}$ follows the prescribed program for every honest party (except for registration of honest users, see below) whereas corrupt parties are controlled by the adversary.

- $\mathcal{DH}_{\text{reg}}$ provides simulated oracles G, H by maintaining lists L_G and L_H of pairs (x, y). Each time a query x to either oracle is made then $\mathcal{DH}_{\text{reg}}$ browses through the corresponding list and searches for an entry (x, y). If such an entry exists then $\mathcal{DH}_{\text{reg}}$ returns y on behalf of the oracle. If no such entry exists then $\mathcal{DH}_{\text{reg}}$ picks y at random (from $\{0,1\}^t$ or $\mathbf{Z}_q$), stores (x, y) in the list and returns y. In addition, $\mathcal{DH}_{\text{reg}}$ can also set values in each list by appending (x, y). If the list becomes inconsistent, i.e., there are entries $(x, y), (x, y')$ for $y \neq y'$, then $\mathcal{DH}_{\text{reg}}$ stops with output $\perp$.

- Each time the adversary instructs an honest user ID_ℓ to register, $\mathcal{DH}_{\text{reg}}$ first waits to receive a value R_ℓ of the certification authority (possibly generated by $\mathcal{DH}_{\text{reg}}$ itself if CA is honest). On behalf of the user $\mathcal{DH}_{\text{reg}}$ selects $c_\ell \xleftarrow{\$} \mathbf{Z}_q$, $v_\ell \xleftarrow{\$} \mathbf{Z}_q^*$ and replies with $S_\ell = (g^b)^{v_\ell}(R_\ell Z^{c_\ell})^{-1}$ for the given g^b. $\mathcal{DH}_{\text{reg}}$ stores the values $(\mathsf{ID}_\ell, (R_\ell, S_\ell), \mathsf{cert}_\ell, v_\ell)$ in a list —we say v_ℓ is associated to $(\mathsf{ID}_\ell, (R_\ell, S_\ell), \mathsf{cert}_\ell)$— and also sets $((R_\ell, S_\ell, \mathsf{ID}_\ell), c_\ell)$ in list L_H. Complete the protocol as the honest user would. Note that corruptions are static, so the user remains honest during this execution.

- For the ℓ-th call $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell, M_{0,\ell}, M_{1,\ell})$ to oracle $\mathcal{UE}$ algorithm $\mathcal{DH}_{\text{reg}}$ rejects invalid queries and otherwise picks $u_\ell \xleftarrow{\$} \mathbf{Z}_q^*$, $\$_\ell \xleftarrow{\$} \{0,1\}^t$ at random and returns $C = ((g^a)^{u_\ell}, \$_\ell)$ for the given g^a. We call u_ℓ associated to the the ℓ-th query.

For the analysis note that $\mathcal{DH}_{\text{reg}}$ stops prematurely because of an inconsistent H-list with probability at most $(Q_H + Q_R)Q_R/2^k$, because each of the Q_R registration steps generates the any of the previous $Q_H + Q_R$ queries with probability at most $1/q$, taking into account the Q_R implicit H-queries in registration steps. Also, up to the point where $\mathcal{A}$ puts the first target query (if at all) the views of the adversary in experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$ and in this simulation are identical. Hence, if there is indeed a target query, then j predicts the target query number correctly with probability at least $1/Q_G$. In addition, the guess i and thus the associated value u_i for the interaction with $\mathcal{UE}$ resulting in the first target query is correct with probability at least $1/Q_{\mathcal{UE}}$.

Now, if $\mathcal{A}$ successfully puts a target query and this query is for a previously registered key $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell)$ of an honest user, then there exists an associated value v_ℓ and $\mathcal{DH}_{\text{reg}}$ outputs

$$Y = X^{1/(u_i v_\ell)} = (R_\ell S_\ell Z^{c_\ell})^{a u_i/(u_i v_\ell)} = (g^{b v_\ell})^{a/v_\ell} = g^{ab}.$$

Hence, this happens with probability at most

$$\Pr[\exists i : \mathsf{TargetQuery}_b^{\text{post}}(i) \wedge \mathsf{RegKey}_b(i)] \;\leq\; Q_{\mathcal{UE}} Q_G \cdot \mathbf{Adv}^{DH}_{\mathcal{EG},\mathcal{DH}_{\text{reg}}}(k) + \frac{(Q_H + Q_R)Q_R}{2^k}\ .$$

Finally, we bound the probability $\epsilon := \Pr[\exists i : \mathsf{TargetQuery}_b^{\text{post}}(i) \wedge \mathsf{UnregKey}_b(i)]$. This is again established in terms of success probability solving the DH problem. However, this time we use an algorithm $\mathcal{DH}_{\text{unreg}}$, which also gets $(\mathcal{G}, q, g, g^a, g^b)$ as input, but uses rewinding techniques for the simulation of $\mathcal{A}$:

- $\mathcal{DH}_{\text{unreg}}$ initially picks index j at random between 1 and the number Q_G of queries to oracle G. $\mathcal{DH}_{\text{unreg}}$ starts to simulate an attack of $\mathcal{A}$ in experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$. In particular, if the adversary asks to corrupt the CA then $\mathcal{DH}_{\text{unreg}}$ immediately stops (recall that we are interested in $\mathsf{UnregKey}_b$ target queries for which the CA remains honest). For an honest authority $\mathcal{DH}_{\text{unreg}}$ sets $pk_{\mathsf{CA}} = Z = g^b$ for the given g^b. $\mathcal{DH}_{\text{unreg}}$ also provides simulated oracles G, H by maintaining two lists L_G and L_H as $\mathcal{DH}_{\text{reg}}$.

- Each time the adversary instructs an honest or corrupt user ID to register, $\mathcal{DH}_{\text{unreg}}$ first picks $r, c \xleftarrow{\$} \mathbf{Z}_q$ and computes $R = g^r Z^{-c}$. We call the value c *pending* for this registration. $\mathcal{DH}_{\text{unreg}}$ sends R on behalf of the CA to the user and waits to retrieve S. Then it sets the oracle value $H(R, S, \mathsf{ID})$ to be the pending value c and replies in the name of the CA with $y = r$.

- For each query $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell, M_{0,\ell}, M_{1,\ell})$ with $pk_\ell = (R_\ell, S_\ell)$ to oracle $\mathcal{UE}$, algorithm $\mathcal{DH}_{\text{unreg}}$ first checks the validity of the request. If the query is invalid then return $\perp$. Else pick $u_\ell \xleftarrow{\$} \mathbf{Z}_q^*$ and $\$ \xleftarrow{\$} \{0,1\}^t$ at random and return $C = ((g^a)^{u_\ell}, \$)$ for the given g^a. Call u_ℓ associated to $(\mathsf{ID}_\ell, pk_\ell, \mathsf{cert}_\ell)$.
- If the adversary puts the j-th query $(\mathsf{ID}, (R,S), \mathsf{cert}, X)$ to oracle G (and $(\mathsf{ID}, (R,S), \mathsf{cert})$ has not been registered yet) then record X and rewind the simulation to the point, where $\mathcal{A}$ has queried oracle H about (R, S, ID) for the first time. If H has never been queried about this value before, then make the query now. Next, start an independent execution from this point on, i.e., with the same adversarial view up to this point (including g^a, g^b) but with independent choices in the sequel. In particular, $\mathcal{DH}_{\text{unreg}}$ removes entries in L_G and L_H made after the rewind point. It replies to the oracle query (R, S, ID) to H with an independent value c' and re-uses all pending values for each registration which has been started but not finished before the rewind point. Also, $\mathcal{DH}_{\text{unreg}}$ renews the choice for index j.
- Eventually, $\mathcal{DH}_{\text{unreg}}$ finishes the second execution, yielding a G-query $(\mathsf{ID}', pk', \mathsf{cert}', X')$, or possibly the output $\perp$. If $(\mathsf{ID}, pk, \mathsf{cert}) = (\mathsf{ID}', pk', \mathsf{cert}')$ and if $c \neq c'$, then look up the associated values $u, u' \in \mathbf{Z}_q^*$ and output $Y = ((X')^{1/u'} X^{-1/u})^{1/(c'-c)}$.

In the first execution, up to the point where $\mathcal{A}$ puts the first target query, the adversary's views of experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$ and of the simulation are the same, and j predicts the right target query with probability $1/Q_G$. Furthermore, the H-list becomes inconsistent with probability at most $(Q_H + Q_R)Q_R/2^k$. Hence, in the first execution we obtain a valid target query for the i-th target and an unregistered key with probability at least

$$\frac{1}{Q_G} \cdot \left(1 - \frac{(Q_H + Q_R)Q_R}{2^k}\right) \cdot \epsilon \, .$$

From now on we condition on this event. For the second execution, define the random variable $\mathsf{RewPoint}$ as the view of the adversary up to the query of (R, S, ID) to H, including the internal coins of $\mathcal{A}$, all queries and answers to oracles, any input data including g^a, g^b and all transcripts. Call $\mathsf{RewPoint}$ *good* iff the probability that $\mathcal{A}$ queries G about the i-th target for unregistered key (R, S, ID) before asking about any other target, given the partial view $\mathsf{RewPoint}$, is at least $\epsilon/2$. Then a standard argument shows $\Pr[\mathsf{RewPoint}\text{ good}] \geq \epsilon/2$.

First note that the probability that the H-list becomes inconsistent in the second execution (because of registration requests) is at most $(Q_H + Q_R)Q_R/2^k$, too. To see this first note that deleting and reselecting H-values for the rewinding creates a consistent view, because the corresponding key $(\mathsf{ID}, (R,S), \mathsf{cert})$ is unregistered and therefore choosing $H(\mathsf{ID}, R, S)$ differently as c' does not affect a pending value. Any other random collision in the H-list happens with the stated probability only.

Given that $\mathsf{RewPoint}$ is good we again get a consistent simulation and target query in the second execution and another good value X' for challenge c' —where $c' \neq c$ with probability at least $1 - 2^{-k}$— with probability at least

$$\frac{1}{Q_G} \cdot \left(1 - \frac{(Q_H + Q_R)Q_R}{2^k}\right) \cdot \left(1 - \frac{1}{2^k}\right) \cdot \frac{\epsilon}{2} \, .$$

The overall probability for the second execution is multiplied by the probability $\epsilon/2$ of $\mathsf{RewPoint}$ being good. Given such two values X, X' with associated values u, c, u', c' we have

$$Y = \left((X')^{1/u'} X^{-1/u}\right)^{1/(c'-c)} = \left((RSZ^{c'})^{au'/u'} (RSZ^{c})^{-au/u}\right)^{1/(c'-c)} = Z^{(c'-c)a/(c'-c)} = g^{ab}$$

and therefore

$$\begin{aligned} \mathbf{Adv}^{DH}_{\mathcal{EG},\mathcal{DH}_{\text{unreg}}}(k) &\geq \frac{1}{Q_G^2} \cdot \frac{\epsilon^3}{4} \left(1 - \frac{(Q_H + Q_R)Q_R}{2^k}\right)^2 \cdot \left(1 - \frac{1}{2^k}\right) \\ &\geq \frac{1}{Q_G^2} \cdot \frac{\epsilon^3}{4} - \frac{(Q_H + Q_R)Q_R + 1}{2Q_G^2 \cdot 2^k} \, . \end{aligned}$$

Re-arranging the equations and putting them together shows security of the simplified scheme against chosen-plaintext attacks. To show security of the original scheme against chosen-ciphertext attacks we have to show that decryption queries do not help the adversary and consider the role of F. Since this proof is very similar to the chosen-plaintext case we only sketch the necessary adaptations.

First, instead of bounding the probability of event $\mathsf{TargetQuery}_b^{\text{pre}}$ by $(Q_G+Q_{\mathcal{UE}})Q_{\mathcal{UE}}/2^k$ here we now obtain the bound $(Q_G + Q_{\mathcal{UE}} + Q_{\mathcal{D}})Q_{\mathcal{UE}}/2^k$ where we take also the at most $Q_{\mathcal{D}}$ implicit G-queries in decryption steps into account. For the bounds for $\Pr[\exists i : \mathsf{TargetQuery}_b^{\text{post}}(i) \wedge \mathsf{RegKey}_b(i)]$ and $\Pr[\exists i : \mathsf{TargetQuery}_b^{\text{post}}(i) \wedge \mathsf{UnregKey}_b(i)]$, we specify how algorithms $\mathcal{DH}_{\text{reg}}$ and $\mathcal{DH}_{\text{unreg}}$ simulate decryption queries in their emulations (with some small error). Then the bounds for the two cases follows as before.

A preliminary observation is that, except with probability $(Q_F + Q_{\mathcal{UE}} + Q_{\mathcal{D}})^2/2^k$, the adversary in experiment $\mathbf{Exp}^{\text{cenc-ind-cpa}}_{\mathsf{CE},\mathcal{A},b}(k)$ here does not submit a valid query $(\mathsf{ID}, pk, \mathsf{cert}, (A, B))$ to the decryption oracle where $A = A_i$ for some of the $\mathcal{UE}$'s answer (A_i, B_i) for the same $(\mathsf{ID}, pk, \mathsf{cert})$. The case that A has been submitted before the challenge ciphertext is generated, is trivially covered by this probability, so we assume that A_i is created afterwards. Then, since decryption requests must be different from the challenge, on one hand $B \neq B_i$. With $a = \log_g A = \log_g A_i = a_i$, on the other hand, it follows $B \oplus G(\mathsf{ID}, pk, \mathsf{cert}, (R_iS_iZ^{c_i})^a) \neq B_i \oplus G(\mathsf{ID}, pk, \mathsf{cert}, (R_iS_iZ^{c_i})^{a_i})$ and therefore the validity check of the decryption process can only succeed if the distinct values collide under F. But this happens with the stated probability only.

Next we explain how the simulators $\mathcal{DH}_{\text{reg}}$ and $\mathcal{DH}_{\text{unreg}}$ deal with decryption requests (without having to know secret keys). But before, we note that $\mathcal{DH}_{\text{reg}}$'s and $\mathcal{DH}_{\text{unreg}}$'s simulation of oracle $\mathcal{UE}$ remains unchanged, i.e., by returning $C = ((g^a)^{u_\ell}, \$_\ell)$ for the ℓ-th query where, this time, $\$_\ell$ is sampled from $\{0,1\}^{t+k}$. In particular, the simulators do not follow the prescribed method by computing $a = F(\mathsf{ID}, pk, \mathsf{cert}, \alpha||M_b)$.

Decryption queries are simulated in the same way for both algorithms. Each simulator maintains another F-list to provide the oracle F, analogously to the the lists for G and H. We again presume that this list is consistent, and we also assume that no F-collisions occur (which happens with probability $2(Q_F + Q_{\mathcal{UE}} + Q_{\mathcal{D}})^2/2^k$ or twice this value, respectively, for the expected two executions $\mathcal{DH}_{\text{unreg}}$ makes). When $\mathcal{A}$ submits a query $(\mathsf{ID}, pk, \mathsf{cert}, C)$ for $C = (A, B)$ to the universal decryption oracle the simulator verifies too that user ID is honest and that there is an entry $(\mathsf{ID}, pk, \mathsf{cert})$ in *RegListPub*. If not then the simulator answers with $\perp$. So suppose in the sequel all decryption queries are valid.

We distinguish between two cases for a value $(\mathsf{ID}, pk, \mathsf{cert}, (A, B))$ submitted to the decryption oracle, depending on the F-list: If there is an entry $((\mathsf{ID}, pk, \mathsf{cert}, \alpha||M), a)$ in the F-list such that $A = g^a$ then the simulator checks that $\alpha||M = B \oplus G(\mathsf{ID}, pk, \mathsf{cert}, (RSZ^c)^a)$ for $pk = (R, S)$ and $c = H(R, S, \mathsf{ID})$ and, if so, returns M. If the equation does not hold then the simulator returns $\perp$. Else, if there is no such query in the list then the simulator simply returns $\perp$. Given that there is such an element in the list, it is unique by our assumption about collision-freeness of F and the simulated decryption generates the same output as the decryption oracle would. If there is no such element then the value A still uniquely determines $a = \log_g A$ and therefore $\alpha||M = B \oplus G(\mathsf{ID}, pk, \mathsf{cert}, (RSZ^c)^a)$. The probability that $F(\mathsf{ID}, pk, \mathsf{cert}, \alpha||M) = a$ is, however, $1/q$ (because this entry is not in the list and the evaluation of F yields a random value in $\mathbf{Z}_q$). The probability that this happens for any of the $Q_{\mathcal{D}}$ many queries to $\mathcal{DH}_{\text{reg}}$ is thus at most $Q_{\mathcal{D}}/2^k$ (and $2Q_{\mathcal{D}}/2^k$ for $\mathcal{DH}_{\text{unreg}}$). A simple calculation now confirms the probabilities. ▌

D Proof of Theorem 4.3

Consider an adversary $\mathcal{A}$ against TCS. We construct adversaries $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_2$ attacking DS such that

$$\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{TCS},\mathcal{A}}(k) \;\leq\; Q_R \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{A}_1}(k) + \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{A}_2}(k) + Q_R \cdot \mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{A}_3}(k) + \frac{Q_R}{2^k}.$$

and the theorem follows by the assumption that DS is a secure digital signature scheme.

In the following, we denote by RegKey the event that in experiment $\mathbf{Exp}^{\text{cs-uf}}_{\mathsf{TCS},\mathcal{A}}(k)$ adversary $\mathcal{A}$ outputs a valid forgery that is with respect to an identity and key that have been properly registered (i.e. case 1. in Definition 4.1), by $\mathsf{UnregKey}$ the event that $\mathcal{A}$ outputs a valid forgery with respect to an identity ID and a key pk such that the key pk has not been priorly registered for user ID (i.e. case 2. from Definition 4.1). Finally, we denote by $\mathsf{CertForge}$ the event that the forgery output by $\mathcal{A}$ is such that the public key pk has been registered for user ID, but it has also been registered for user ID': that is the event described in case 3. of Definition 4.1 occurs, and the event described in case 2. of the same definition does not occur.

It is immediate that

$$\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{TCS},\mathcal{A}}(k) \;\leq\; \Pr[\,\mathsf{RegKey}\,] + \Pr[\,\mathsf{UnregKey}\,] + \Pr[\,\mathsf{CertForge}\,] \tag{1}$$

and the theorem follows by bounding the terms on the right hand side.

Adversary $\mathcal{A}_1$ is against the scheme DS, so it takes as input a public key pk^* and has access to a signing oracle $\mathcal{S}(sk^*, \cdot)$. Here, (pk^*, sk^*) is a pair of keys generated by running the key generation algorithm of DS on input I. Adversary $\mathcal{A}_1$ runs adversary $\mathcal{A}$ as a subroutine, and simulates for $\mathcal{A}$ its interaction in experiment $\mathbf{Exp}^{\text{cs-uf}}_{\mathcal{A},\mathsf{TCS}}(k)$. In particular it plays the role of all honest parties, including that of the CA if it does not get corrupted by $\mathcal{A}$.

Adversary $\mathcal{A}_1$ starts its execution by selecting a random position i between 1 and Q_R and then starts execution adversary $\mathcal{A}$. If $\mathcal{A}$ decides to corrupt the CA, $\mathcal{A}_1$ receives pk_{CA} from $\mathcal{A}$. Otherwise, $\mathcal{A}_1$ generates the key by itself, and in all future registration requests it plays the role of the CA.

Adversary $\mathcal{A}_1$ handles all registration requests for honest users by playing their role (and that of the CA if the latter is not corrupted). The transcripts of the registration protocol are handed back to $\mathcal{A}$. There is one exception: for the i'th registration request, the public key that is registered for user ID_i is set to pk^*, i.e. the key that $\mathcal{A}_1$ has as input. Notice that $\mathcal{A}$ can sign the challenge message M of the CA by submitting message $(0||M)$ to its oracle.

Valid signature requests are answered as follows: all such requests other than for signatures by user ID_i, are computed by $\mathcal{A}_1$ itself since it posses the appropriate signing keys. For a valid signature request $(\mathsf{ID}_i, pk^*, \mathsf{cert}_i, M)$ adversary $\mathcal{A}_1$ obtains σ by querying M to its own signature oracle, i.e. $\sigma \xleftarrow{\$} (sk^*, (1||M))$.

At some point $\mathcal{A}$ finishes its execution and outputs a forgery $(\mathsf{ID}, pk, \mathsf{cert}, M, \sigma)$. If the forgery output by $\mathcal{A}$ is valid and $pk = pk^*$ then $\mathcal{A}_1$ outputs $(1||M, \sigma)$ as its own forgery.

We now analyze the advantage of $\mathcal{A}_1$. The simulation of the environment of $\mathcal{A}$ is perfect, so, adversary $\mathcal{A}$ outputs a valid forgery $(\mathsf{ID}, pk, \mathsf{cert}, M, \sigma)$ for the key pk registered by honest user ID with probability $\Pr[\,\mathsf{RegKey}\,]$. With probability $\frac{1}{Q_R}$, adversary $\mathcal{A}$ had set this registered key to pk^*, so if this is the case, M, σ are such that $\mathcal{V}_1(pk^*, 1||M, \sigma) = 1$. Therefore, $(1||M, \sigma)$ is a valid forgery for A_1 provided that M has never been queried to signing oracle of $\mathcal{A}_1$. Such a query only occurs if $\mathcal{A}$ queries $(\mathsf{ID}_i, pk^*, \mathsf{cert}_i, M)$ to its signing oracle, but in this case $(\mathsf{ID}, pk, \mathsf{cert}, M)$ is not a valid forgery for $\mathcal{A}$. We conclude that:

$$\mathbf{Adv}^{\text{ds-uf}}_{\mathsf{DS},\mathcal{A}_1}(k) \geq \frac{1}{Q_R} \cdot \Pr[\,\mathsf{RegKey}\,]\,. \tag{2}$$

We now continue with the construction of $\mathcal{A}_2$. It takes as input a public key pk^* and has access to a signing oracle $\mathcal{S}(sk^*, \cdot)$, with (pk^*, sk^*) a pair of keys generated via $(pk^*, sk^*) \xleftarrow{\$} \mathcal{SK}(1^k)$. Adversary $\mathcal{A}_3$ executes adversary $\mathcal{A}$ as a subroutine and emulates its environment by intercepting and answering its queries. The simulation will be as if the secret key of the CA is sk^*, the key of the signing oracle of $\mathcal{A}$, so if $\mathcal{A}$ requests to corrupt the CA then $\mathcal{A}_3$ aborts its execution. Otherwise, $\mathcal{A}_2$ will play the role of the CA in all future registration requests and the signatures of the CA are produced by queries to the oracle of $\mathcal{A}_2$. So, when $\mathcal{A}$ requires the registration of an (honest or corrupt) user ID_l, $\mathcal{A}_2$ plays the role of the CA: if the user is corrupt it receives from $\mathcal{A}$ a public key pk_l to be registered for ID_l, otherwise it generates a pair (pk_l, sk_l) by itself. Signatures of the CA needed in the registration process are obtained from the oracle of $\mathcal{A}_2$.

Valid signature requests can be answered trivially by $\mathcal{A}_2$ since it possesses all secret keys of the honest users.

When $\mathcal{A}$ outputs a forgery $(\mathsf{ID}, pk, \mathsf{cert}, M, \sigma)$, adversary $\mathcal{A}_2$ outputs its attempted forgery $((\mathsf{ID}, pk), \mathsf{cert})$.

It is clear that the simulation that $\mathcal{A}_2$ provides to $\mathcal{A}$ is perfect. Assume that $\mathcal{A}$ produces a valid forgery $(\mathsf{ID}, pk, \mathsf{cert}, M, \sigma)$. with respect to an unregistered key pk for user ID (notice that in this case adversary $\mathcal{A}$ is not allowed to corrupt the CA). This event happens with probability $\Pr[\,\mathsf{UnregKey}\,]$. We argue that in this case the forgery of $\mathcal{A}_2$ is valid. Indeed, we have that $\mathcal{V}_1(pk^*, (\mathsf{ID}, pk), \mathsf{cert}) = 1$ since $(\mathsf{ID}, pk, \mathsf{cert}, M, \sigma)$ is a valid forgery for $\mathcal{A}$. Moreover, the message (ID, pk) has not been queried to the signing oracle since such a request occurs only when the key pk is registered for identity ID. Therefore we have that:

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{A}_2}(k) \geq \Pr[\,\mathsf{UnregKey}\,]\,. \tag{3}$$

Finally, we construct adversary $\mathcal{A}_3$ against DS. It runs adversary $\mathcal{A}$ as a subroutine and wins against DS, whenever the event $\mathsf{CertForge}$ occurs in the execution of $\mathcal{A}$. Algorithm $\mathcal{A}_3$ takes as input a public key pk^* and has access to a signing oracle $\mathcal{S}(sk^*, \cdot)$ with (pk^*, sk^*) generated by the key generation algorithm of DS. It simulates the environment of $\mathcal{A}$ exactly as adversary $\mathcal{A}_3$ does. In particular it selects a random index $i \in [Q_R]$, and registers pk^* as the public key of user ID_i; notice that $\mathcal{A}_3$ can easily answer the challenge M_i sent by the CA during the registration protocol by submitting $0||M$ to its signing oracle. Following this execution, for the rest of registration requests, adversary $\mathcal{A}_3$ tracks the key pk_l that each user ID_l attempts to register. If the execution of the protocol was successful and $pk_l = pk^*$ then let M_l be the challenge message sent by the CA to ID_l and σ_l the signature that is returned. Then adversary $\mathcal{A}_3$ outputs as its attempted forgery message $(0||M_l, \sigma_l)$.

The success probability of $\mathcal{A}_3$ is as follows: if event $\mathsf{CertForge}$ occurs in the execution of $\mathcal{A}$, then there exist two indices l_0 and l_1 such that the same key pk is registered for both identities ID_{l_0} and ID_{l_1}, and with the registration of ID_{l_0} taking place before that of ID_{l_1}. With probability $1/Q_R$, the index i selected by $\mathcal{A}_3$ equals l_0 and therefore with this probability the key registered for these two identities is pk^*. Since the registration of ID_{l_1} is successful, signature σ_l is a valid signature with respect to pk^* on message $0||M_{l_1}$. This is a valid forger if this message has not been queried to the oracle of $\mathcal{A}_3$. However, the only other message of the form $0||M$ that is queried to the oracle is $0||M_{l_0}$, and the two messages are equal only with probability $\frac{1}{2^k}$. We therefore have that:

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS},\mathcal{A}_3} \geq \frac{1}{Q_R} \cdot \Pr\left[\,\mathsf{CertForge}\,\right] - \frac{1}{2^k}$$

The desired result follows by putting together the above inequalities.

E Discrete Logarithm Assumption and Proof of Theorem 4.5

Discrete Logarithm Assumption. We say that the parameter generator $\mathcal{SG}$ in Construction 4.4 generates *DL-secure groups* if for any expected polynomial-time algorithm A the following is negligible in k:

$$\mathbf{Adv}^{DL}_{\mathcal{SG},A}(k) = \Pr\left[\,(\mathcal{G}, q, g, G, H) \xleftarrow{\$} \mathcal{SG}(1^k);\; a \xleftarrow{\$} \mathbf{Z}_q \;:\; A(\mathcal{G}, q, g, g^a) = a\,\right]\,,$$

Proof of Theorem 4.5 We prove our theorem by showing that there exist adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ for solving the discrete logarithm problem such that:

$$\begin{aligned}\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k) \leq & Q_G^2 \cdot \mathbf{Adv}^{DL}_{\mathcal{SG},\mathcal{B}_1}(k) + Q_G^2 \cdot \frac{Q_S(Q_S+Q_G) + Q_R(Q_R+Q_H)+1}{2^k} + \\ & + Q_G^2 Q_H^2 \cdot \left\{ \mathbf{Adv}^{DL}_{\mathcal{SG},B_2}(k) + \left[\frac{4}{Q_G^2 Q_H^2}\left(\mathbf{Adv}^{DL}_{\mathcal{SG},B_2}(k) + \frac{3}{2^k}\right)\right]^{\frac{1}{3}} + 2 \cdot \frac{Q_R(Q_R+Q_H)+1}{2^k} \right\}\end{aligned}$$

We denote by RegKey the event that $\mathcal{A}$ outputs a valid forgery that contains the key registered by some honest user. Notice that we do not require that the identity in the forgery be the one for which the key was registered. We denote by $\mathsf{UnregKey}$ the event that $\mathcal{A}$ outputs a valid forgery with respect to an unregistered key. Clearly,

$$\mathbf{Adv}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k) = \Pr\left[\,\mathsf{RegKey}\,\right] + \Pr\left[\,\mathsf{UnregKey}\,\right].$$

We complete the proof by bounding the terms on the right hand side using the advantage of two adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ in computing discrete logarithms.

Algorithm $\mathcal{B}_1$ takes as input a description of a group $(\mathcal{G}, q, g)$ and a challenge $S = g^s$ with s selected at random in $\mathbf{Z}_q$. It runs a black-box simulation of $\mathcal{A}$'s attack and uses well-known rewinding techniques to extract the discrete log of S. It operates as follows:

- $\mathcal{B}_1$ selects a random index $i \xleftarrow{\$} [Q_G]$, where Q_G is the maximal number of queries that $\mathcal{A}$ makes to oracle G in the experiment $\mathbf{Exp}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k)$. $\mathcal{B}_1$ sets the initial parameters to $(\mathcal{G}, q, g, G, H)$, where G and H are random oracles that $\mathcal{B}_1$ simulates for $\mathcal{A}$ as usual by maintaining two lists L_G, L_H. $\mathcal{B}_1$ runs attacker $\mathcal{A}$ as a subroutine, by mimicking its environment in experiment $\mathbf{Exp}^{\text{cs-uf}}_{\mathsf{CS},\mathcal{A}}(k)$. If adversary $\mathcal{A}$ requests to corrupt the CA then $\mathcal{A}$ provides pk_{CA} and plays the role of the CA in all executions of the registration protocol.

If the CA stays honest, then $\mathcal{B}_1$ generates $pk_{\mathsf{CA}} = g^z$ with $z \xleftarrow{\$} \mathbf{Z}_q$ and plays the role of the CA in all executions of the registration protocol.

- When the adversary makes its l'th registration query for honest identity ID_l, $\mathcal{B}_1$ runs the protocol on behalf of the honest party as follows. First, it receives a message R_l from the CA. Chooses random $c_l, t_l \xleftarrow{\$} \mathbf{Z}_q$ and sets: $S_l = (R_l Z^{c_l})^{-1} S^{t_l}$ and adds $((\mathsf{ID}_l, R_l, S_l), c_l)$ to the list L_H. It then sends S_l to $\mathcal{A}$ and obtains in return y_l. The public key of ID_l is set to (R_l, S_l).
- When adversary makes a valid signature query $(\mathsf{ID}, (R, S), \mathsf{cert}, M)$, i.e. $(\mathsf{ID}, R, S) = (\mathsf{ID}_l, R_l, S_l)$ that occurred in the registration of ID_l, $\mathcal{B}_1$ computes a valid signature on M as follows: it selects at random $d \in \mathbf{Z}_q$, selects at random $B \in \mathcal{G}$ and computes $A = B(RSZ^c)^{-d}$ and $c = H(R, S, \mathsf{ID})$. Then it adds the pair $((\mathsf{ID}, A, M), d)$ to the list L_G (i.e. it sets $G(\mathsf{ID}, A, M) = d$) and returns to $\mathcal{A}$ the signature (A, B).
- At some point, adversary $\mathcal{A}$ stops and outputs a forgery of the form: $(\mathsf{ID}, (R, S), \epsilon, M, (A, B))$. If $(R, S) \neq (R_l, S_l)$ for some R_l, S_l that occurred during the registration of identity ID_l, then $\mathcal{B}_1$ aborts its execution. If message (ID, A, M) is not the i'th query of $\mathcal{A}$ to oracle G or lists L_G, L_H are inconsistent then $\mathcal{B}_1$ also aborts. Otherwise:
- It rewinds the execution of $\mathcal{A}$ up to the point where it had queried the message $(\mathsf{ID}, A||M)$ to the oracle G (if this query did not occur, then $\mathcal{B}_1$ makes it now). Let d_1 be the response that $\mathcal{B}_1$ returned to $\mathcal{A}$ during the first execution. Adversary $\mathcal{B}_1$ selects a new value $d_2 \xleftarrow{\$} \mathbf{Z}_q$, returns d_2 to $\mathcal{A}$ and continues the execution from this point on. If $\mathcal{A}$ outputs another forgery $(\mathsf{ID}', (R', S'), \epsilon, M', (A', B'))$ such that $(\mathsf{ID}', R', S', M', A') = (\mathsf{ID}, R, S, M, A)$ and the lists L_H, L_G are still consistent, then adversary $\mathcal{B}_1$ terminates the entire execution and outputs $s = (B - B')(t_l d_1 - t_l d_2)^{-1}$ if $d_1 \neq d_2$. Otherwise repeat this last step.

First we argue that if the execution of $\mathcal{B}_1$ enters the loop and finishes for $d_1 \neq d_2$, then the output of $\mathcal{B}_1$ is $\log S$. Indeed, at the end of the execution $\mathcal{B}_1$ has obtained two valid forgeries with respect to some identity ID, public key (R_l, S_l) registered during the execution some message and some message M. Since the signatures are both valid, we have that:

$$g^B = A(R_l S_l Z^{H(\mathsf{ID}, R_l, S_l)})^{d_1} \quad \text{and} \quad g^{B'} = A(R_l S_l Z^{H(\mathsf{ID}, R_l, S_l)})^{d_2}$$

where, by construction, $S_l = (R_l Z^{H(\mathsf{ID}_l, R_l, S_l)})^{-1} g^{t_l s}$. It follows that $B = \log A + t_l s d_1$ and $B' = \log A + t_l s d_2$ and the conclusion follows by simple algebraic manipulation.

It follows from standard considerations that $\mathcal{B}_1$ makes two executions of $\mathcal{A}$ on the average. We now compute the success probability of $\mathcal{B}_1$. A simple counting argument shows that there exists some index $i_0 \in [Q_G]$ such that $\Pr[\,\mathsf{RegKey}(i_0)\,] \geq \frac{1}{Q_G} \cdot \Pr[\,\mathsf{RegKey}\,]$. From now on we condition on the event that $\mathcal{B}_1$'s guess i equals this index i_0. This happens with probability $1/Q_G$. Under this condition $i = i_0$, if we start looping then we always obtain a second forgery, and with probability at least $1 - 1/2^k$ this will be for $d_1 \neq d_2$. Since the probability of a success in the first execution and therefore of running the rewind strategy is given by the probability of $\mathsf{RegKey}(i_0)$ minus the probability for inconsistent lists L_G, L_H, we get:

$$\mathbf{Adv}^{DL}_{\mathcal{SG},\mathcal{B}_1}(k) \;\geq\; \frac{1}{Q_G} \cdot \Pr[\,\mathsf{RegKey}(i_0)\,] - \frac{1}{2^k} - \frac{Q_S(Q_S + Q_G)}{2^k} - \frac{Q_R(Q_R + Q_H)}{2^k}.$$

and

$$\Pr[\,\mathsf{RegKey}\,] \;\leq\; Q_G^2 \cdot \mathbf{Adv}^{DL}_{\mathcal{SG},\mathcal{B}_1}(k) + Q_G^2 \cdot \frac{Q_S(Q_S + Q_G) + Q_R(Q_R + Q_H) + 1}{2^k}.$$

We now bound the probability $\Pr[\,\mathsf{UnregKey}\,]$ that during experiment $\mathbf{Exp}^{\mathsf{cs\text{-}uf}}_{\mathsf{CS},\mathcal{A}}(k)$ adversary $\mathcal{A}$ outputs a valid forgery with respect to a key that has not been already registered. We show how to use $\mathcal{A}$ to build an adversary $\mathcal{B}_2$ that solves the discrete logarithm problem.

Adversary $\mathcal{B}_2$ takes as input a description of a group $(\mathcal{G}, q, g)$ and a challenge $Z = g^z$ with z selected at random in $\mathbf{Z}_q$. It runs adversary $\mathcal{A}$ as a subroutine, performing several rewindings of its execution. Roughly, our goal is to find the discrete logarithm of Z from the verification equation $g^B = A(RSZ^c)^d$. Unlike in the previous case the key R, S is not registered here, and it is likely that d depends on c (if not, extraction becomes much easier). Hence, we have to use nested rewindings, where we use the same extraction strategy as for $\mathcal{B}_1$ for the inner loop to extract the logarithm of $A = g^{-B}(RSZ^c)^d$ by getting two forgeries for different d-challenges. Each time this terminates in a constant number of repetitions on the average, and the overall running time of $\mathcal{B}_2$ therefore follows analogously as for $\mathcal{B}_1$. We next try to do this extraction twice for different c-challenges such that for each of these two solutions we can "peel off" the outer d-layer to solve for Z. We only repeat the outer loop twice in order to not run into trouble with the expected running times, but at the cost of decreasing our success probability.

We now give an description of how adversary $\mathcal{B}_2$ simulates the environment of $\mathcal{A}$ without the rewindings; the rewinding strategy follows the strategy we outlined above.

- $\mathcal{B}_2$ selects at random two indices $i \xleftarrow{\$} [Q_G]$ and $j \xleftarrow{\$} [Q_H]$. $\mathcal{B}_2$ sets the initial parameters to $(\mathcal{G}, q, g, G, H)$ where G and H are oracles that $\mathcal{B}_2$ simulates as usual by maintaining two lists L_G, L_H. In particular, we denote by $\alpha_H \leq Q_R(Q_R + Q_H)/2^k$ the probability of getting an inconsistent list L_H during one execution.
- Adversary $\mathcal{B}_2$ simulates attacks of $\mathcal{A}$ against an instantiation of CS in which pk_{CA}, the public key of the CA, is set to Z (the challenge that $\mathcal{B}_2$ has as input). If $\mathcal{A}$ decides to corrupt the CA then the $\mathcal{B}_1$ aborts its execution.
- When adversary $\mathcal{A}$ makes its l'th registration query for honest identity ID_l, algorithm $\mathcal{B}_2$ selects $s_l, c_l, y_l \xleftarrow{\$} \mathbf{Z}_q$. It then computes $R_l = g^y Z^{-c_l}$, adds the pair $((R_l, S_l, \mathsf{ID}_l), c_l)$ to the list L_H and then hands $R_l, (R_l, y_l)$ to the adversary as transcript of the registration it required; If the identity ID_l is corrupted, adversary $\mathcal{B}_2$ proceeds as above, with the only difference that S_l is obtained from the adversary;
- When adversary $\mathcal{A}$ makes a valid signature request $(\mathsf{ID}, pk, \mathsf{cert}, M)$, the adversary simply computes the signature since it knows the secret keys of all honest users. This particularly implies that the G-list L_G never becomes inconsistent.

As explained above, we omit a formal description of how $\mathcal{B}_2$ rewinds the executions. Eventually, $\mathcal{B}_2$ either aborts prematurely or gets four forgeries $(\mathsf{ID}^i, (R^i, S^i), \epsilon, M^i, (A^i, B^i))$, for $i = 1, 2, 3, 4$ such that $\mathsf{ID}^1 = \mathsf{ID}^2 = \mathsf{ID}^3 = \mathsf{ID}^4$, $R^1 = R^2 = R^3 = R^4$, $S^1 = S^2 = S^3 = S^4$, $A^1 = A^2$ and $A^3 = A^4$, which satisfy the equations:

$$B^1 = \log A^1 + (\log S^1 + \log R^1 + c_1 z) \cdot d_1 \ ; \quad B^2 = \log A^2 + (\log S^2 + \log R^2 + c_1 z) \cdot d_2$$
$$B^3 = \log A^3 + (\log S^3 + \log R^3 + c_2 z) \cdot d_3 \ ; \quad B^4 = \log A^4 + (\log S^4 + \log R^4 + c_2 z) \cdot d_4$$

for $c_1 \neq c_2$, $d_1 \neq d_2$, $d_3 \neq d_4$. A simple calculation shows that this yields $z = \log Z$.

In the following, we denote by $\mathsf{UnregKey}(i, j, G)$ the event that $\mathcal{A}$ outputs a valid forgery with respect to (i, j) and the ith query to G is made before the jth query to H. Also, by $\mathsf{UnregKey}(i, j, H)$ denotes the event that $\mathcal{A}$ outputs a valid forgery with respect to (i, j), and the jth query to H is made before the jth query to G. So, for any $(i, j) \in [Q_G] \times [Q_H]$ we have that:

$$\Pr[\, \mathsf{UnregKey}(i, j) \,] = \Pr[\, \mathsf{UnregKey}(i, j, G) \,] + \Pr[\, \mathsf{UnregKey}(i, j, H) \,] \, .$$

It is easy to see that there exists a pair of indices $(i_0, j_0) \in [Q_G] \times [Q_H]$ such that:

$$\Pr[\, \mathsf{UnregKey}(i_0, j_0, G) \,] + \Pr[\, \mathsf{UnregKey}(i_0, j_0, H) \,] \geq \frac{1}{Q_G \cdot Q_H} \cdot \Pr[\, \mathsf{UnregKey} \,] \tag{4}$$

Suppose that the choice (i, j) of $\mathcal{B}_2$ coincides with (i_0, j_0) as above. This occurs with probability $\frac{1}{Q_G Q_H}$.

We sketch the two possibilities how $\mathcal{B}_2$ finds the discrete logarithm of z. First, if $\mathcal{A}$ makes the i-th query to G before the j-th query to H, i.e., the value d is the same for both challenges $c_1 \neq c_2$, then $\mathcal{B}_2$ gets two equations

$$g^{B^1} = A(RSZ^{c_1})^d = A(RS)^d Z^{dc_1} \qquad \text{and} \qquad g^{B^2} = A(RSZ^{c_2})^d = A(RS)^d Z^{dc_2}$$

from which $\mathcal{B}_2$ can easily compute z. This succeeds for different challenges $c_1 \neq c_2$ and a consistent list L_H and thus with probability at least $\frac{1}{Q_G \cdot Q_H} \cdot \Pr[\, \mathsf{UnregKey}(i_0, j_0, G) \,] - \frac{1}{2^k} - \alpha_H$.

The other possibility is that $\mathcal{A}$ makes the i-th query to G after the j-th query to H. For i, j let $\epsilon(i, j) = \Pr[\, \mathsf{UnregKey}(i, j, H) \,] - \alpha_H$ bound the probability that the first execution of $\mathcal{A}$ as a subroutine of $\mathcal{B}$ ends with a valid forgery with respect to (i, j), query j to H is made before query i to G and adversary $\mathcal{B}_2$ does not abort due to an inconsistency in the list L_H. The analysis here is very similar to the case of the ElGamal-based encryption where we also used two rewinding executions to extract the Diffie-Hellman solution (only here we have nested another rewinding inside). It analogously follows that $\mathcal{B}_2$ succeeds in finding solutions for different challenges $c_1 \neq c_2$, $d_1 \neq d_2$, $d_3 \neq d_4$ with probability at least $\frac{1}{4 Q_G Q_H} \cdot \epsilon(i_0, j_0)^3 - \frac{3}{2^k}$. Given that, $\mathcal{B}_2$ can derive z as described above.

Since adversary $\mathcal{B}_2$ succeeds when either case above occurs we have that $\mathbf{Adv}^{DL}_{\mathcal{SG},\mathcal{B}_2}(k)$ is a greater than both success probabilities. From Equation (4) we thus conclude that $\Pr[\,\mathsf{UnregKey}\,]$ is bounded by

$$Q_G^2 Q_H^2 \cdot \left\{ \mathbf{Adv}^{DL}_{\mathcal{SG},B_2}(k) + \frac{1}{2^k} + \left[\frac{4}{Q_G^2 Q_H^2} \left(\mathbf{Adv}^{DL}_{\mathcal{SG},B_2}(k) + \frac{3}{2^k} \right) \right]^{\frac{1}{3}} + \left(1 + \frac{1}{Q_G Q_H} \right) \cdot \alpha_H \right\}$$

Using the bounds on $\Pr[\,\mathsf{RegKey}\,]$ and $\Pr[\,\mathsf{UnregKey}\,]$ the desired result follows. ▮