Practical Signing-Right Revocation*

Michael Till Beck[1], Stephan Krenn[2], Franz-Stefan Preiss[3], and Kai Samelin[3,4]

[1] Ludwig-Maximilians-Universität München, Munich, Germany
michael.beck@ifi.lmu.de
[2] AIT Austrian Institute of Technology GmbH, Vienna, Austria
stephan.krenn@ait.ac.at
[3] IBM Research – Zurich, Rüschlikon, Switzerland
{frp,ksa}@zurich.ibm.com
[4] Technische Universität Darmstadt, Darmstadt, Germany

Abstract. One of the key features that must be supported by every modern PKI is an efficient way to determine (at verification) whether the signing key had been revoked. In most solutions, the verifier periodically contacts the certificate authority (CA) to obtain a list of blacklisted, or whitelisted, certificates. In the worst case this has to be done for every signature verification. Besides the computational costs of verification, after revocation *all* signatures under the revoked key become invalid. In the solution by Boneh et al. at USENIX '01, the CA holds a share of the private signing key and contributes to the signature generation. After revocation, the CA simply denies its participation in the interactive signing protocol. Thus, the revoked user can no longer generate valid signatures. We extend this solution to also cover privacy, non-trusted setups, and time-stamps. We give a formal definitional framework, and provide elegantly simple, yet provably secure, instantiations from efficient standard building blocks such as digital signatures, commitments, and partially blind signatures. Finally, we provide extensions to our scheme.

Changes to Prior Version. Clearly, in the privacy games the adversary's chance of success needs to negligibly close to $\frac{1}{2}$, and not 0. This was only a typo, and has no impact on the schemes, games, or proofs.

1 Introduction

Digital signatures [24] provide meaningful security as long as the signing key stays secret. However, in the real-world, signing keys can be compromised very easily, e.g., through hacker attacks, lost hardware tokens, or simply by accident. Furthermore, it is often required to revoke signing rights, e.g., when an employee leaves a company. Consequently, deployed solutions such as X.509, and related standards, always allow for revocation of certificates [12, 19]. Here, two main approaches (and potentially combinations thereof) are deployed. First, in

*This work was supported by the European Commission through grant agreement numbers 321310 (PERCY), 644962 (PRISMACLOUD), and 653454 (CREDENTIAL).

a *white-list* approach, the certificate authority (CA) vouches for the fact that a given certificate is not revoked. Alternatively, the CA can publish a *black-list* containing all revoked certificates. Now, a verifier directly rejects a signature if the used key has been black-listed. Thus, if one requires up-to-date information, this means that the lists must be retrieved *for every signature verification*, causing a high — and sometimes too high — computational and communicational overhead. Thus, in either case, the verifiers contact the CA to determine whether a given certificate is still valid. Thus, every verifier must periodically update the published lists in both approaches to have meaningful security guarantees.

Moreover, as noted by Boneh et al. [9], these *total* revocation mechanisms have several drawbacks. For example, as mentioned previously, to check the revocation status of a given certificate, the verifier must have access to an up-to-date certificate revocation list (CRL), or the CA has to be queried for each signature verification. The latter may not be possible, however, as the verifier may not have a network connection, or communication is too costly. Furthermore, if a certificate is revoked, all signatures corresponding to the contained public key pk, including the ones that were generated honestly, become invalid after revocation. However, it is desirable that all signatures under a secret key sk that were generated prior to the corruption of sk (or prior to the revocation of the corresponding certificate) remain valid, while the generation of new signatures under sk is not possible. For example, consider Spider-Man sending the message m="I admit that you, Iron Man, are more powerful than me."[1] Clearly, if m is signed with Spider-Man's secret key sk, Iron Man can publish the signature to prove to the public that he is more powerful than Spider-Man. However, if Spider-Man revokes his certificate, the signature becomes invalid, and there is no way for Iron Man to prove that the statement is valid. This is because if the secret key sk is corrupted, it cannot be proven that Iron Man is not the adversarial party generating *new* bogus signatures on behalf of Spider-Man. The problem is that signatures are not associated with their generation time, i.e., a new signature is as good as an old one, if no further means such as time-stamping services are involved. Thus, all signatures have to be revoked in this setting. Refer to Gutmann for additional problems of PKIs in their current form [25].

Our Contribution. We address the aforementioned unsatisfactory situation by introducing the notion of CA-assisted signature generation with time-stamping, message privacy, and non-trusted setup. In a nutshell, our scheme requires that a *partially* trusted CA blindly signs the message m in question plus potentially a time-stamp (and some other technical values such as keys, etc.), while a trusted setup is not required. In particular, the CA checks whether the corresponding user's pk is revoked, and signs m only if pk not revoked. The signature generated by the CA is then additionally signed with a standard digital signature scheme by the user. Both signatures are subsequently sent to, and verified, by the verifier. Signatures can be generated as long as the corresponding public key is not

[1] For all Spider-Man fans: please reverse the roles of Spider-Man and Iron Man.

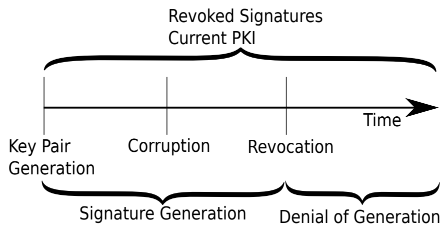


Fig. 1. Revocation of Certificates.

revoked. Therefore, all generated signatures remain valid after revocation as the CA simply stops assisting the signer after the key gets revoked.

While technically being relatively simple, our construction solves most of the mentioned problems, and, interestingly enough, is even more efficient than most deployed solutions, as the CAs are no longer queried for each verification. Moreover, we want that our solution can be added "on-top" of the existing PKI, i.e., the users do not require new keys, while the existing method can co-exist. If a time-stamping authority and traditional revocation lists are naïvely used to solve the problem, the signing process needs to be interactive similar to our construction (because the time-stamp needs to be bound to the signed message). However, our solution does not require *any* interactivity upon verification, which is needed in the naïve solution in order to update revocation information. Moreover, our construction paradigm is elegantly simple, yet versatile. We show how it can easily be extended to cover additional application scenarios. Interestingly, when one tries to close the remaining gap between corruption and revocation (cf. Fig. 1), the resulting construction becomes very similar to the naïve solution again (cf. Sect. 4.1). However, in this case it is easy to see that interactivity is needed for signing (because of the time-stamp) as well as for verification (to check whether a signature key has been revoked "into the past").

Even though the CA is only partially trusted, we do not lose anything, as some kind of trust anchor is always required for a PKI anyway. Our approach actually requires less trust: for white-lists, the CA learns if signatures for a specific public key are verified, while in a black-list approach everyone sees which certificates are revoked. In our solution, the CA only learns when a signature is generated, which happens less frequently. Morever, we have a fall-back mode, which allows to revert to standard signatures.

State-of-the-Art. The idea to let a (semi-)trusted entity such as a CA also contribute to signature generation has been introduced by Boneh et al. [9] and Rivest [34], but neither present a formalization. The approach by Boneh et al. is based on standard 2-out-of-2 threshold signatures [8, 21]. In particular, the secret key sk is split between the CA and the signer. The server denies its contribution to signature generation, if the presented certificate is marked as revoked. However, their approach requires trusted setup (the suggested mitigation strategy of using a distributed key generation algorithm here is too inefficient in practice), new keys for each participant, and cannot add time-stamps to generated signatures. Moreover, an adversarial server may also learn the message to be signed, i.e., in contrast to our solution no privacy guarantees are given to the user. A similar approach is deployed in anonymous credentials such as Identity Mixer [12, 16], where the credential holder proves that it is not revoked at presentation of the credential, e.g., using accumulators [6, 13, 20, 33]. Here, the prover has to prove knowledge of a witness (in zero-knowledge) such that its revocation handle is contained in the accumulator, which resembles a white-list approach. Clearly, the witnesses have to be updated for each revocation, while credentials are, compared to digital signatures, only valid once at presentation.

Blind signatures have been introduced by Chaum [17]. In a nutshell, blind signatures allow an external entity to receive a signature σ on a message m (of its own choice) such that the signer learns nothing about the message m, and cannot link a signing transcript to the final signature. Chaum's work was later formalized and proven secure [4, 27]. Later, constructions in the standard model [14], based on different assumptions other than RSA [8], additional security guarantees [22], but also some impossibility results were published [23]. The initial idea was also extended to cover some form of partial blindness, where the signature is issued on the blinded message m, but also some public information info known to both parties [1, 18]. These partially blind signatures are mostly used to prevent misuse of blind signatures. We use this possibility to bind a signature to a public key, and add time-stamps.

There is also the notion of certificate-less cryptography [2, 26]. In our case we only require a certificate, there are no *ephemeral* keys, and no identity management. However, the ideas are very similar, and can thus be seen as related. Likewise, the concept of *virtual smart cards* [15] is related. However, in contrast to our approach, the additional server is not trusted by outsiders and the signer has to provide an additional password. Moreover, for an outsider (i.e., verifier), a signature generated with their scheme is indistinguishable from a traditional signature. This is not what we want, i.e., a verifier must be able to decide whether a signature was generated using out method.

There are also other primitives which may be used in our context, e.g., threshold signatures [21], proxy signatures [29], server-assisted signatures [7], multi signatures [5], aggregate signatures [10], or sanitizable signatures [3, 11, 28]. However, all these approaches do not offer privacy (i.e., they reveal the message to the server) without further modifications. We therefore chose to use primitives which directly give us the required guarantees.

2 Preliminaries and Building Blocks

Notation. $\lambda \in \mathbb{N}$ denotes our security parameter. All algorithms implicitly take 1^λ as an additional input. We write $a \leftarrow A(x)$ if a is assigned the output of algorithm A with input x. An algorithm is efficient if it runs in probabilistic polynomial time (ppt) in the length of its input. The algorithms may return a special error symbol $\perp \notin \{0,1\}^*$, denoting an exception. For the remainder of this paper, all algorithms are ppt if not explicitly mentioned otherwise. If we have a list, we require that we have an injective, and efficiently reversible encoding mapping the list to $\{0,1\}^*$. If we have a set S, we assume a lexicographical ordering on the elements. A message space $\mathcal{M}$, and the randomness space $\mathcal{R}$, may implicitly depend on a corresponding public key. If not otherwise stated, we assume that $\mathcal{M} = \{0,1\}^*$ to reduce unhelpful boilerplate notation. A function $\nu : \mathbb{N} \to [0,1]$ is *negligible*, if it vanishes faster than every inverse polynomial, i.e., $\forall k \in \mathbb{N}$, $\exists n_0 \in \mathbb{N}$ such that $\nu(n) \leq n^{-k}$, $\forall n > n_0$.

Non-Interactive Commitments. Non-interactive commitment schemes allow one party to commit itself to a value without revealing it. Later, the committing party can give some opening information to the receiver, which can then "open" the commitment.

Definition 1 (Non-Interactive Commitments). *A non-interactive commitment scheme* COM *consists of three ppt algorithms* $\{\mathsf{ParGen}, \mathsf{Commit}, \mathsf{Open}\}$*, such that:*

ParGen. *This algorithm takes as input a security parameter λ and outputs the public parameters* pp*, i.e.,* $\mathsf{pp} \leftarrow \mathsf{ParGen}(1^\lambda)$.

Commit. *This algorithm takes as input a message m and outputs a commitment C together with corresponding opening information O, i.e.,* $(C, O) \leftarrow \mathsf{Commit}(\mathsf{pp}, m)$.

Open. *This deterministic algorithm takes as input a commitment C with corresponding opening information O and outputs message $m \in \mathcal{M}$, i.e.,* $m \leftarrow \mathsf{Open}(\mathsf{pp}, C, O)$.

Definition 2 (Binding). *A non-interactive commitment scheme is* binding*, if for all ppt adversaries $\mathcal{A}$ there is a negligible function $\nu(\cdot)$ such that*

$$\Pr\left[\begin{array}{c}\mathsf{pp} \leftarrow \mathsf{ParGen}(1^\lambda), (C^*, O^*, O'^*) \leftarrow \mathcal{A}(\mathsf{pp}), m \leftarrow \mathsf{Open}(\mathsf{pp}, C^*, O^*), \\ m' \leftarrow \mathsf{Open}(\mathsf{pp}, C^*, O'^*) : m \neq m' \quad \wedge \quad m \neq \perp \quad \wedge \quad m' \neq \perp\end{array}\right] \leq \nu(\lambda).$$

Definition 3 (Perfectly Hiding). *A non-interactive commitment scheme is* perfectly hiding*, if for all unbounded adversaries $\mathcal{A}$ we have*

$$\Pr\left[\begin{array}{c}(\mathsf{pp}, m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda), b \leftarrow \{0,1\}, \\ (C, O) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_b), b^* \leftarrow \mathcal{A}(C, \mathsf{state}) : \\ b = b^*\end{array}\right] - \frac{1}{2} = 0.$$

We say that a commitment scheme COM is correct, if for all $\lambda \in \mathbb{N}$, all $\mathsf{pp} \leftarrow \mathsf{ParGen}(1^\lambda)$, for all messages m, for all $(C, O) \leftarrow \mathsf{Commit}(\mathsf{pp}, m)$, we have $\mathsf{Open}(\mathsf{pp}, C, O) = m$.

A non-interactive commitment scheme COM is secure, if it is correct, binding, and perfectly hiding. An example for such a commitment-scheme are Pedersen-Commitments [32]. We stress that the message space of the Pedersen-Commitments can be extended using collision-resistant hash-functions.

```
**Experiment** $\mathsf{eUNF-CMA}_{\mathcal{A}}^{\mathsf{DSIG}}(\lambda)$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$
    $\mathcal{Q} \leftarrow \emptyset$
    $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}'(\mathsf{sk},\cdot)}(\mathsf{pk})$
        where oracle $\mathsf{Sign}'$ on input $m$:
            set $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
            return $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
    return 1, if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = \texttt{true} \ \wedge \ m^* \notin \mathcal{Q}$
    return 0
```

Fig. 2. Unforgeability

Digital Signatures. Digital signatures allow the holder of a secret key sk to sign a message m, while with knowledge of the corresponding public key pk everyone can verify whether a given signature was actually endorsed by the signer.

Definition 4 (Digital Signatures). *A standard digital signature scheme* DSIG *consists of three algorithms* $\{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}\}$ *such that:*

KGen**.** *The algorithm* KGen *outputs the public and private key of the signer, where* λ *is the security parameter:* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$.

Sign**.** *The algorithm* Sign *gets as input the secret key* sk*, and the message* $m \in \mathcal{M}$ *to sign. It outputs a signature* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$.

Verify**.** *The algorithm* Verify *outputs a decision bit* $d \in \{\texttt{false}, \texttt{true}\}$*, indicating if the signature* σ *is valid, w.r.t.* pk*, and* m*:* $d \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$.

For each DSIG we require the correctness properties to hold. In particular, we require that for all $\lambda \in \mathbb{N}$, for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, for all $m \in \mathcal{M}$ we have $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = \texttt{true}$. This definition captures perfect correctness.

Unforgeability. Now, we define unforgeability of digital signature schemes, as given in [24]. In a nutshell, we require that an adversary $\mathcal{A}$ cannot (except with negligible probability) come up with a signature σ^* for a *new* message m^*. The adversary $\mathcal{A}$ can adaptively query for signatures on messages of its own choice.

Definition 5 (Unforgeability). *A signature scheme* DSIG *is unforgeable, if for any ppt adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\Pr[\mathsf{eUNF-CMA}_{\mathcal{A}}^{\mathsf{DSIG}}(1^\lambda) = 1] \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 2.*

We call a digital signature scheme DSIG secure, if it is correct, and unforgeable.

Partially Blind Signatures. Blind Signatures [17, 27] allow the holder of a secret key to sign a message m for a second entity. The signer does not learn what message it signs, and also cannot link a signature generation transcript against the final signature. Partially Blind Signatures [1] also allow to add some piece of "public" information, known to both parties, to the final signature. Note, for the following definition, we omit the case where some "public parameters" are generated, as it depends on the underlying scheme whether this algorithm is required. An extension is straightforward.

Definition 6 (Partially Blind Signatures). *A partially blind signature scheme* BSIG *consists of two algorithms* $\{\mathsf{KGen}, \mathsf{Verify})$*, and an interactive protocol* $\langle\mathcal{B},\mathcal{U}\rangle$ *such that:*

KGen**.** *The algorithm* KGen *outputs the public and private key of the signer, where* λ *is the security parameter:* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*.*

$\langle\mathcal{B},\mathcal{U}\rangle$**.** *The algorithm* $\langle\mathcal{B},\mathcal{U}\rangle$ *is interactive. The user* $\mathcal{U}$ *receives input* m*, public information* info*, and* pk*. The signer* $\mathcal{B}$ *inputs the secret key* sk*, and some string* info*, while the user* $\mathcal{U}$ *inputs a public key* pk*, a message* m*, and the string* info*. At the end of the protocol, only the user* $\mathcal{U}$ *receives a signature* σ*, while* $\mathcal{B}$ *receives nothing. We denote this as* $(\perp, \sigma) \leftarrow \langle\mathcal{B}(\mathsf{sk}, \mathsf{info}), \mathcal{U}(\mathsf{pk}, m, \mathsf{info})\rangle$*. We write* $\langle\cdot, \mathcal{U}(\cdot,\cdot,\cdot)\rangle^\infty$ *if the adversary plays the role of the signer* $\mathcal{B}$*, can start a new signing session with* $\mathcal{U}$ *as often as it wants to, and can arbitrarily schedule the interactions. Likewise, if we write* $\langle\mathcal{B}(\cdot,\cdot),\cdot\rangle^1$*, the adversary acts as the user, and can interact with the signer only once. We also require that every entity is able to decide to what step of which "session" a given protocol message corresponds, and also when a given "signing session" is finished, and was successful. In particular, we say a signing session is finished once* $\mathcal{B}$ *sends its last message to* $\mathcal{U}$*, and* $\mathcal{U}$ *can actually extract a valid signature.*

Verify**.** *The algorithm* Verify *outputs a decision bit* $d \in \{\mathit{false}, \mathit{true}\}$*, indicating the validness of the signature* σ*, w.r.t.* pk*,* info*, and* m*:* $d \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \mathsf{info}, \sigma)$*.*

For each BSIG we require the correctness properties to hold. In particular, we require that for all $\lambda \in \mathbb{N}$, for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, for all $m \in \mathcal{M}$, for all $\mathsf{info} \leftarrow \{0,1\}^*$ we have $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{info}, \sigma) = \mathtt{true}$, where σ is taken from $(\perp, \sigma) \leftarrow \langle\mathcal{B}(\mathsf{sk}, \mathsf{info}), \mathcal{U}(\mathsf{pk}, m, \mathsf{info})\rangle$. This captures perfect correctness.

We now introduce the security requirements needed for our construction.

Unforgeability. Now, we define unforgeability of partially blind signature schemes, as given in [1, 31], but adjusted for our notation. In a nutshell, we require that an adversary $\mathcal{A}$ cannot (except with negligible probability) come up with more signatures for different message/information pair (m, info) than successful, i.e., completed, signing queries. Note, the adversary can interleave signing queries.

```
$\textbf{Experiment}$ $\mathsf{omUNF} - \mathsf{CMA}_{\mathcal{A}}^{\mathsf{BSIG}}(\lambda)$
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$
  $((m_1, \sigma_1, \mathsf{info}_1), \ldots, (m_\ell, \sigma_\ell, \mathsf{info}_\ell)) \leftarrow \mathcal{A}^{\langle \mathcal{B}(\mathsf{sk}, \cdot, \cdot), \cdot \rangle^\infty}(\mathsf{pk})$
  return 1, if $\forall i \in \{1, 2, \ldots, \ell\} : \mathsf{Verify}(\mathsf{pk}, m_i, \mathsf{info}_i, \sigma_i) = \mathtt{true}$
    and oracle $\langle \mathcal{B}(\mathsf{sk}, \cdot), \cdot \rangle$ finished less than $\ell$ times, and
    all $(m_i, \mathsf{info}_i)$ are pairwise distinct
  return 0
```

Fig. 3. Unforgeability

```
$\textbf{Experiment}$ $\mathsf{Blindness}_{\mathcal{A}}^{\mathsf{BSIG}}(\lambda)$
  $(\mathsf{pk}^*, \{m_0, m_1\}, \mathsf{info}, \mathsf{state}_1) \leftarrow \mathcal{A}(1^\lambda)$
  $b \leftarrow \{0, 1\}$
  $\mathsf{state}_2 \leftarrow \mathcal{A}^{\langle \cdot, \mathcal{U}_0(\mathsf{pk}^*, m_b, \mathsf{info}) \rangle^1, \langle \cdot, \mathcal{U}_1(\mathsf{pk}^*, m_{1-b}, \mathsf{info}) \rangle^1}(\mathsf{state}_1)$
  let $\sigma_0$, and $\sigma_1$ denote the output of $\mathcal{U}_0$, and $\mathcal{U}_1$
  If $\sigma_0 = \perp \vee \sigma_1 = \perp$, let $\sigma \leftarrow \perp$
  Else, set $\sigma \leftarrow (\sigma_b, \sigma_{1-b})$
  $a \leftarrow \mathcal{A}(\mathsf{state}_2, \sigma)$
  return 1, if $a = b$
  return 0
```

Fig. 4. Blindness

Definition 7 (Unforgeability). *A signature scheme* BSIG *is unforgeable, if for any ppt adversary $\mathcal{A}$ there exists a negligible function ν such that* $\Pr[\mathsf{omUNF} - \mathsf{CMA}_{\mathcal{A}}^{\mathsf{BSIG}}(1^\lambda) = 1] \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 3.*

Note, we define "weak" unforgeability, i.e., once a signature for a given message/information pair (m, info) becomes known, the adversary may be able to derive new signatures.

Blindness. Now, we define blindness of partially blind signature schemes, derived from [31]. In a nutshell, we require that an adversary $\mathcal{A}$ cannot (except with negligible probability) decide what message is signed, and cannot link a signing transsript against the final signature. This must even be true, if it can generate the public key, chose the messages to be signed, and also the public string info.

Definition 8 (Blindness). *A partially blind signature scheme* BSIG *is blind, if for any ppt adversary $\mathcal{A}$ there exists a negligible function ν such that* $\left|\Pr[\mathsf{Blindness}_{\mathcal{A}}^{\mathsf{BSIG}}(1^\lambda) = 1] - \frac{1}{2}\right| \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 4.*

We call a partially blind signature scheme BSIG secure, if it is correct, unforgeable, and blind. Jumping ahead, we use the public information to embed the current time-stamp, and the signer's public key into the signature.

3 CA-Assisted Signatures

We now introduce CA-Assisted Signatures. As already discussed in the introduction, the main idea is that a CA helps generating a signature.

3.1 Syntax

In the following we now give a formal specification of the algorithms and their interfaces in such schemes. We require that each party has access to a common clock which is synchronized across all parties. In practice, this can be realized, e.g., by using the Network Time Protocol [30], and checking that the time-stamp is in an acceptable range, say, e.g., 30 seconds.

Definition 9 (CA-Assisted Signatures). *A CA-assisted digital signature scheme* CASIG *consists of four algorithms* $\{\mathsf{KGen}_u, \mathsf{KGen}_c, \mathsf{Revoke}, \mathsf{Verify}\}$ *and one interactive protocol* $\langle \mathcal{CA}, \mathcal{U} \rangle$ *such that:*

KGen_u**.** *The algorithm* KGen_u *outputs the public and private key of each user, where* λ *is the security parameter:* $(\mathsf{pk}_u, \mathsf{sk}_u) \leftarrow \mathsf{KGen}(1^\lambda)$.

KGen_c**.** *The algorithm* KGen_c *outputs the public and private key of a* $\mathcal{CA}$*, where* λ *is the security parameter:* $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{KGen}(1^\lambda)$.

$\langle \mathcal{CA}, \mathcal{U} \rangle$**.** *The protocol* $\langle \mathcal{CA}, \mathcal{U} \rangle$ *is interactive. The user* $\mathcal{U}$ *receives input* m*,* pk_s*,* time*, and* sk_u*. The* $\mathcal{CA}$ *inputs the secret key* sk_c*,* time*, and* pk_u*. At the end of the protocol, only the user* $\mathcal{U}$ *receives a signature* σ *(which may be* $\perp$ *for a revoked user), while* $\mathcal{CA}$ *receives nothing:* $(\perp, \sigma) \leftarrow \langle \mathcal{CA}(\mathsf{sk}_s, \mathsf{pk}_u, \mathsf{time})), \mathcal{U}(\mathsf{sk}_u, \mathsf{pk}_u, m, \mathsf{time}) \rangle$*. As for partially blind signatures, we assume that each party knows to which signing session, and which protocol step a received message belongs to, and is successful.*

Revoke**.** *The algorithm* Revoke *allows to revoke a given public key* pk_u*. In a nutshell, the CA no longer agrees to start a signing protocol for revoked* pk_u*. Thus, revocation of a* pk_u *does not affect already ongoing signing sessions for this* pk_u*. This algorithm outputs nothing.*

Verify**.** *The algorithm* Verify *outputs a decision bit* $d \in \{\mathtt{false}, \mathtt{true}\}$*, indicating the validness of the signature* σ*, with respect to* pk_c*,* pk_s*,* time*, and* m*:* $d \leftarrow \mathsf{Verify}(\mathsf{pk}_c, \mathsf{pk}_u, m, \mathsf{time}, \sigma)$*.*

3.2 Definitional Framework for CA-Assisted Signatures

We now define the formal requirements for CA-assisted signatures. In a nutshell, those are correctness, unforgeability against malicious users and CAs, and blindness/privacy against CAs and outsiders.

Correctness. As usual, we require correctness of any CASIG. In particular, we require that with overwhelming probability in the security parameter it holds that $\mathsf{Verify}(\mathsf{pk}_c, \mathsf{pk}_u, m, \mathsf{time}, \sigma) = \mathtt{true}$, where $(\mathsf{pk}_u, \mathsf{sk}_u) \leftarrow \mathsf{KGen}_u(1^\lambda)$, $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{KGen}_c(1^\lambda)$, $m \in \mathcal{M}$, $\mathsf{time} \in \mathbb{N}$, $(\perp, \sigma) \leftarrow \langle \mathcal{CA}(\mathsf{sk}_c, \mathsf{pk}_s, \mathsf{time}), \mathcal{U}(\mathsf{sk}_s, \mathsf{pk}_u, m, \mathsf{time})\rangle$, and pk_u was not revoked *before the signature generation request*. The probability space is here given by all random coins in all involved algorithms. The scheme is said to be *perfectly correct*, if σ verifies correctly with probability 1.

Unforgeability. Unforgeability of CA-assisted signatures covers two aspects. On the one hand, a malicious user must not be able to fake signatures of the CA. On the other hand, a malicious CA must not be able to impersonate a user. Together, those two definitions clearly also imply that an outsider is not able to forge any valid signatures.

For signer unforgeability, we allow an adversary to obtain arbitrarily many signatures on arbitrary messages, and public keys, of its choice. Furthermore, for every signature, the adversary may define the current time (except that it may not turn back the time). Also, he can generate and revoke user keys at convenience. Similarly to Def. 7, the adversary now wins if it can output more message/signature pairs than he queried from the oracle; furthermore, each of those pairs must only verify for a public key and time that have been used in a signing query. Finally, signatures may only verify if the corresponding user public key has not been revoked before starting the respective signing session. For simplicity, we define that if a signing oracle is tagged as "non-called", if the corresponding public key was revoked before the current time. In the case that revocation and signing were done at the very same point in time, we do not consider the signature a forgery even if the revocation request was submitted first in the experiment; one the one hand, this is a purely academic issue anyways, and on the other hand "before" and "after" do not have any semantics within a fixed point in time.

Definition 10 (Signer Unforgeability). *A CA-assisted signature scheme* CASIG *is signer unforgeable, if for any ppt adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\Pr[\mathsf{seUNF} - \mathsf{CMA}_{\mathcal{A}}^{\mathsf{CASIG}}(1^\lambda) = 1] \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 5.*

Complementary to signer unforgeability, we also require that the CA cannot generate valid signatures for a specific user without its contribution. We therefore let the adversary (controlling the CA) obtain arbitrarily many signatures for a user public key pk_u, where again $\mathcal{A}$ has full control over time. The adversary now wins if he can output a signature on a message that was not asked for that specific define point in time. This definition is similar to the standard definition of unforgeability, cf. Def. 5.

Note that as before, the adversary is allowed to interleave signing queries. Further note that the given definition is only presented in its weak formulation, i.e., the adversary is allowed to output fresh signatures for message/time

```
**Experiment** $\mathsf{seUNF-CMA}_{\mathcal{A}}^{\mathsf{CASIG}}(\lambda)$
  $(\mathsf{sk}_c, \mathsf{pk}_c) \leftarrow \mathsf{KGen}_c(1^\lambda)$
  $\mathsf{time} \leftarrow 0$
  $((m_1, \sigma_\ell, \mathsf{time}_1, \mathsf{pk}_1), \ldots, (m_\ell, \sigma_\ell, \mathsf{time}_\ell, \mathsf{pk}_\ell)) \leftarrow \mathcal{A}^{\langle \mathcal{CA}(\mathsf{sk}_c, \cdot, \mathsf{time}), \cdot \rangle^\infty, \mathsf{Timestamp}(\cdot), \mathsf{Revoke}(\cdot)}(\mathsf{pk}_c)$
  where oracle $\mathsf{Timestamp}$ on input $\mathsf{time}'$:
    if $\mathsf{time}' \leq \mathsf{time}$, ignore
    let $\mathsf{time} \leftarrow \mathsf{time}'$
  return 1, if $\forall i \in \{1, 2, \ldots, \ell\} : \mathsf{Verify}(\mathsf{pk}_c, \mathsf{pk}_i, m_i, \mathsf{time}_i, \sigma_i) = \mathtt{true}$
    and oracle $\langle \mathcal{CA}(\mathsf{sk}, \cdot, \cdot), \cdot \rangle$ finished less than $\ell$ times, and
    all $(m_i, \mathsf{time}_i, \mathsf{pk}_i)$ are pairwise distinct
  return 1, if $\mathsf{Verify}(\mathsf{pk}_c, \mathsf{pk}_1, m_1, \mathsf{time}_1, \sigma_1) = \mathtt{true}$,
    and $\mathsf{pk}_1$ was revoked before $\mathsf{time}_1$
  return 0
```

Fig. 5. Signer Unforgeability

```
**Experiment** $\mathsf{ceUNF-CMA}_{\mathcal{A}}^{\mathsf{CASIG}}(\lambda)$
  $(\mathsf{sk}_u, \mathsf{pk}_u) \leftarrow \mathsf{KGen}_u(1^\lambda)$
  $\mathsf{time} \leftarrow 0$
  $(m^*, \sigma^*, \mathsf{time}^*, \mathsf{pk}^*) \leftarrow \mathcal{A}^{\langle \cdot, \mathcal{U}(\mathsf{sk}_u, \cdot, \cdot, \mathsf{time}) \rangle^\infty, \mathsf{Timestamp}(\cdot)}(\mathsf{pk}_u)$
  where oracle $\mathsf{Timestamp}$ on input $\mathsf{time}'$:
    if $\mathsf{time}' \leq \mathsf{time}$, ignore
    let $\mathsf{time} \leftarrow \mathsf{time}'$
  return 1, if $\mathsf{Verify}(\mathsf{pk}^*, \mathsf{pk}_u, m^*, \mathsf{time}^*, \sigma^*) = \mathtt{true}$,
    and oracle $\langle \cdot, \mathcal{U}(\mathsf{sk}_u, \cdot, \cdot, \cdot) \rangle$ was never queried for $(\mathsf{pk}^*, m^*, \mathsf{time}^*)$.
  return 0
```

Fig. 6. CA Unforgeability

pairs for which it obtained honest signatures. Extending the definition to strong unforgeability is straightforward.

Definition 11 (CA Unforgeability). *A CA-assisted signature scheme* CASIG *is* CA unforgeable, *if for any ppt adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\Pr[\mathsf{ceUNF-CMA}_{\mathcal{A}}^{\mathsf{CASIG}}(1^\lambda) = 1] \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 6.*

Blindness. Blindness is concerned with the privacy of the user towards the CA. While a secure CA-assisted signature scheme must satisfy both aspects of unforgeability, blindness comes in two flavors giving different privacy guarantees.

```
$\textbf{Experiment}\ \mathsf{CA-Blindness}_{\mathcal{A}}^{\mathsf{CASIG}}(\lambda)$
    $(\mathsf{sk}_u, \mathsf{pk}_u) \leftarrow \mathsf{KGen}_u(1^\lambda)$
    $(\mathsf{pk}^*, \{m_0, m_1\}, \mathsf{time}, \mathsf{state}_1) \leftarrow \mathcal{A}(\mathsf{pk}_u)$
    $b \leftarrow \{0,1\}$
    $\mathsf{state}_2 \leftarrow \mathcal{A}^{\langle \cdot, \mathcal{U}_0(\mathsf{sk}_u, \mathsf{pk}^*, m_b, \mathsf{time})\rangle^1, \langle \cdot, \mathcal{U}_1(\mathsf{sk}_u, \mathsf{pk}^*, m_{1-b}, \mathsf{time})\rangle^1, \mathsf{Revoke}(\cdot)}(\mathsf{state}_1)$
    let $\sigma_0$, and $\sigma_1$ denote the output of $\mathcal{U}_0$, and $\mathcal{U}_1$.
    If $\sigma_0 = \perp \vee \sigma_1 = \perp$, let $\sigma \leftarrow \perp$.
    Else, set $\sigma \leftarrow (\sigma_b, \sigma_{b-1})$
    $a \leftarrow \mathcal{A}(\mathsf{state}_2, \sigma)$
    return 1, if $a = b$
    return 0
```

Fig. 7. CA Blindness

The first flavor, called *CA blindness*, is similar in spirit to Def. 8. There, the CA (controlled by the adversary) may trigger signing protocols on two messages of its choice in a random order, gets the resulting signatures, and then needs to link the transcripts to the messages.

In the second flavor, called *CA weak-blindness*, we only require that the adversary does not learn which message it signed. In particular, the adversary does not gain access to the signatures, and may only trigger a single signing query. It is easy to see that CA blindness implies CA weak-blindness, but not vice versa. The decision which level of blindness/privacy is required must be made on a case-to-case basis, depending on the concrete use case.

Similar to Def. 8, the adversary is restricted to a single interaction with each oracle in our blindness definitions. However, blindness against multiple protocol runs directly follows from a simple hybrid argument.

Definition 12 (CA Blindness). *A CA-assisted signature scheme* CASIG *is* CA blind, *if for any ppt adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\left|\Pr[\mathsf{CA-Blindness}_{\mathcal{A}}^{\mathsf{CASIG}}(1^\lambda) = 1] - \frac{1}{2}\right| \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 7.*

Definition 13 (CA Weak-Blindness). *A CA-assisted signature scheme* CASIG *is* weakly CA-blind, *if for any ppt adversary* $\mathcal{A}$ *there exists a negligible function* ν *such that* $\left|\Pr[\mathsf{CA-WBlindness}_{\mathcal{A}}^{\mathsf{CASIG}}(1^\lambda) = 1] - \frac{1}{2}\right| \leq \nu(\lambda)$. *The corresponding experiment is depicted in Fig. 8.*

We call a CA-assisted signature scheme CASIG *secure and (weakly) blind*, if it is correct, signer unforgeable, CA unforgeable, and CA (weakly) blind.

```
**Experiment** $\mathsf{CA}-\mathsf{WBlindness}_{\mathcal{A}}^{\mathsf{CASIG}}(\lambda)$
    $(\mathsf{sk}_u, \mathsf{pk}_u) \leftarrow \mathsf{KGen}_u(1^\lambda)$
    $(\mathsf{pk}^*, \{m_0, m_1\}, \mathsf{time}, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pk}_u)$
    $b \leftarrow \{0,1\}$
    $a \leftarrow \mathcal{A}^{\langle \cdot, \mathcal{U}(\mathsf{sk}_u, \mathsf{pk}^*, m_b, \mathsf{time})\rangle^1, \mathsf{Revoke}(\cdot)}(\mathsf{state})$
    return 1, if $a = b$
    return 0
```

Fig. 8. Weak CA Blindness

4 Constructions

We now show how to come up with constructions achieving what we want. First, we present a generic construction, which, depending on the used building blocks, achieves weaker, or stronger resp., privacy notions. We stress that our reductions are tight, i.e., we have no reduction losses, and thus omit a probability analysis in the proofs.

Generic Construction Idea. Let us introduce the generic idea of our construction first. We then give two different derivations of the generic constructions, but instantiated with different building blocks. Both constructions offer the same unforgeability guarantees, but offer a different level of privacy.

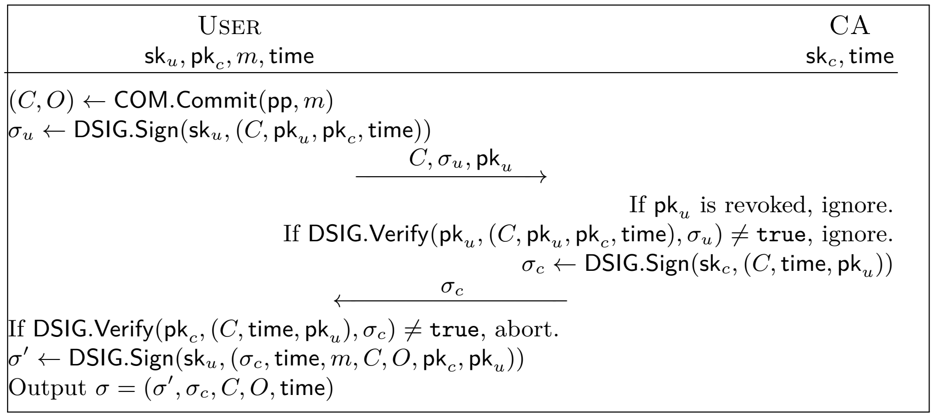
In a nutshell, we let a CA contribute to signature generation, but only if the public key of the requester is not revoked at the time time of the signature request. The CA can then also add some additional information to the final signature such as certificates, and the like. However, from a privacy point of view, it is also required that the CA does not learn which messages are signed, which reflects blindness.

On the one hand, we let the signer commit to a message, and the let the CA sign this commitment, and the signer's public key, if, and only if, the given public key is not revoked. The user, on the other hand, creates an additional signature around the received signature from the CA to protect against bogus CAs. Clearly, there is no joint setup, and thus key generation can be done offline, which is not possible in current schemes. We stress that revoking a public key is simply sending the CA a message "My pk has been revoked", possibly containing a proof of knowledge, which is not necessarily zero-knowledge.

Note, the parties do not need to communicate using a secure channel.

Construction 1 (Weakly-Blind Construction) *Let* $\mathsf{CASIG} := (\mathsf{KGen}_u, \mathsf{KGen}_c, \langle \mathcal{CA}, \mathcal{U}\rangle, \mathsf{Revoke}, \mathsf{Verify})$ *such that:*

KGen_u**.** *Generate a key-pair of a standard digital signature scheme, i.e., return* $(\mathsf{pk}_u, \mathsf{sk}_u) \leftarrow \mathsf{DSIG.KGen}(1^\lambda)$.



User $\mathsf{sk}_u, \mathsf{pk}_c, m, \mathsf{time}$ — CA $\mathsf{sk}_c, \mathsf{time}$

$(C, O) \leftarrow \mathsf{COM.Commit}(\mathsf{pp}, m)$
$\sigma_u \leftarrow \mathsf{DSIG.Sign}(\mathsf{sk}_u, (C, \mathsf{pk}_u, \mathsf{pk}_c, \mathsf{time}))$

$$\xrightarrow{C, \sigma_u, \mathsf{pk}_u}$$

If pk_u is revoked, ignore.
If $\mathsf{DSIG.Verify}(\mathsf{pk}_u, (C, \mathsf{pk}_u, \mathsf{pk}_c, \mathsf{time}), \sigma_u) \neq \mathtt{true}$, ignore.
$\sigma_c \leftarrow \mathsf{DSIG.Sign}(\mathsf{sk}_c, (C, \mathsf{time}, \mathsf{pk}_u))$

$$\xleftarrow{\sigma_c}$$

If $\mathsf{DSIG.Verify}(\mathsf{pk}_c, (C, \mathsf{time}, \mathsf{pk}_u), \sigma_c) \neq \mathtt{true}$, abort.
$\sigma' \leftarrow \mathsf{DSIG.Sign}(\mathsf{sk}_u, (\sigma_c, \mathsf{time}, m, C, O, \mathsf{pk}_c, \mathsf{pk}_u))$
Output $\sigma = (\sigma', \sigma_c, C, O, \mathsf{time})$

Fig. 9. CA-Assisted Signing With Weak Blindness

KGen_c**.** *Generate a key-pair of a standard digital signature scheme* $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{DSIG.KGen}(1^\lambda)$*, and the public parameters* $\mathsf{pp} \leftarrow \mathsf{COM.ParGen}(1^\lambda)$ *of a commitment scheme. Return* $((\mathsf{pk}_c, \mathsf{pp}), \mathsf{sk}_c)$*.*

$\langle \mathcal{CA}, \mathcal{U} \rangle$**.** *See Fig. 9.*

Verf**.** *To verify a signature* $\sigma = (\sigma', \sigma_c, C, O, \mathsf{time})$ *w.r.t.* m*,* pk_c*, and* pk_u*, check that* $m = \mathsf{COM.Open}(\mathsf{pp}, C, O)$*, and* $\mathsf{DSIG.Verify}(\mathsf{pk}_c, (C, \mathsf{time}, \mathsf{pk}_u), \sigma_c) = \mathtt{true}$*,* $\mathsf{DSIG.Verify}(\mathsf{pk}_u, (\sigma_c, \mathsf{time}, m, C, O, \mathsf{pk}_c, \mathsf{pk}_u), \sigma') = \mathtt{true}$*. If all checks pass, output* $\mathtt{true}$*, and* $\mathtt{false}$ *otherwise.*

Theorem 1. *If* DSIG *and* COM *are secure, then our construction is secure and weakly blind.*

Proof. Correctness follows from inspection. Thus, we only consider signer unforgeability, CA unforgeability, weak blindness. We prove each property on its own.

Signer Unforgeability. Let $\mathcal{A}$ be an adversary which can break the signer unforgeability of our construction. We can then construct an adversary $\mathcal{B}$ which either breaks the binding property of COM, or the unforgeability of the signature scheme DSIG used by the CA. Assume that there is a signature σ on the message $(\sigma_c, \mathsf{time}, m, C, O, \mathsf{pk}_c, \mathsf{pk}^*)$, where σ_c is a signature on the message (C, time), but also a signature σ' for $(\sigma_c, \mathsf{time}', m', C, O, \mathsf{pk}_c, \mathsf{pk}'^*)$, where $(m, \mathsf{time}, \mathsf{pk}^*) \neq (m', \mathsf{time}', \mathsf{pk}'^*)$. Hence, we have two different messages which "are in" the same commitment. Clearly, this breaks the binding property of the commitment scheme used. In the second case, i.e., there is a new commitment C' for $(m, \mathsf{time}) \neq (m', \mathsf{time}')$ never signed by the CA, the adversary must have been able to forge a signature σ'_c. This also accounts for a revoked public key. In both cases a reduction for $\mathcal{B}$ is trivial, and therefore omitted.

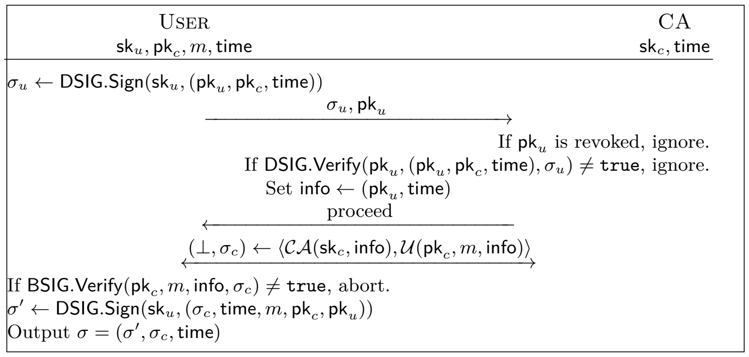

User
$\mathsf{sk}_u, \mathsf{pk}_c, m, \mathsf{time}$
CA
$\mathsf{sk}_c, \mathsf{time}$
$\sigma_u \leftarrow \mathsf{DSIG.Sign}(\mathsf{sk}_u, (\mathsf{pk}_u, \mathsf{pk}_c, \mathsf{time}))$
σ_u, pk_u
If pk_u is revoked, ignore.
If $\mathsf{DSIG.Verify}(\mathsf{pk}_u, (\mathsf{pk}_u, \mathsf{pk}_c, \mathsf{time}), \sigma_u) \neq \mathtt{true}$, ignore.
Set $\mathsf{info} \leftarrow (\mathsf{pk}_u, \mathsf{time})$
proceed
$(\perp, \sigma_c) \leftarrow \langle \mathcal{CA}(\mathsf{sk}_c, \mathsf{info}), \mathcal{U}(\mathsf{pk}_c, m, \mathsf{info}) \rangle$
If $\mathsf{BSIG.Verify}(\mathsf{pk}_c, m, \mathsf{info}, \sigma_c) \neq \mathtt{true}$, abort.
$\sigma' \leftarrow \mathsf{DSIG.Sign}(\mathsf{sk}_u, (\sigma_c, \mathsf{time}, m, \mathsf{pk}_c, \mathsf{pk}_u))$
Output $\sigma = (\sigma', \sigma_c, \mathsf{time})$

Fig. 10. CA-Assisted Signing with Blindness

CA Unforgeability. This case is trivial as well. If the adversary $\mathcal{A}$ can come up with a signature on a message $(\sigma_c, \mathsf{time}, m, C, O, \mathsf{pk}^*, \mathsf{pk}_u)$, where $(m, \mathsf{time}, \mathsf{pk}^*)$ was never signed, then it can break the unforgeability of the used signature scheme. Again, a reduction is straightforward.

CA Weak-Blindness. Trivial, as COM is perfectly hiding, and therefore σ_u is independent of m, which is the only information sent to the CA, i.e., $\mathcal{A}$. ☐

Construction 2 (Blind Construction) *Let* $\mathsf{CASIG}' := (\mathsf{KGen}_u, \mathsf{KGen}_c, \langle \mathcal{CA}, \mathcal{U} \rangle, \mathsf{Revoke}, \mathsf{Verify})$ *such that:*

KGen_u**.** *Generate a key-pair of a standard digital signature scheme, i.e., return* $(\mathsf{pk}_u, \mathsf{sk}_u) \leftarrow \mathsf{DSIG.KGen}(1^\lambda)$.

KGen_c**.** *Generate a key-pair of a partially blind signature scheme, i.e., return* $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{BSIG.KGen}(1^\lambda)$.

$\langle \mathcal{CA}, \mathcal{U} \rangle$**.** *See Fig. 10.*

Verf**.** *To verify a signature* $\sigma = (\sigma', \sigma_c, \mathsf{time})$ *w.r.t.* m, pk_c, *and* pk_u, *check that* $\mathsf{DSIG.Verify}(\mathsf{pk}_c, (\sigma_c, \mathsf{time}, m, \mathsf{pk}_c, \mathsf{pk}_u), \sigma') = \mathtt{true}$, *and* $\mathsf{BSIG.Verify}(\mathsf{pk}_c, m, (\mathsf{pk}_u, \mathsf{time}), \sigma_c) = \mathtt{true}$. *If all checks pass, output* $\mathtt{true}$, *and* $\mathtt{false}$ *otherwise.*

Theorem 2. *If* DSIG *and* BSIG *are secure, then our construction is secure and blind.*

Proof. Again, correctness follows by inspection. It remains to prove CA unforgeability, signer unforgeability, and blindness.

Signer Unforgeability. Let $\mathcal{A}$ be an adversary which can break the signer unforgeability of our construction. We can then construct an adversary $\mathcal{B}$ which breaks the unforgeability of the partially blind signature scheme. $\mathcal{B}$ receives pk from the BSIG to forge, and embeds the received pk into the public key pk_c. It simply follows the protocol, and uses its own oracle to get signatures. If a given pk_i is revoked, $\mathcal{B}$ no longer accepts new signing sessions. Eventually, $\mathcal{A}$ outputs

$((m_1, \sigma_1, \mathsf{info}_1, \mathsf{pk}_1), \ldots, (m_\ell, \sigma_\ell, \mathsf{time}_\ell, \mathsf{pk}_\ell))$. Clearly, if pk_1 was revoked, $\mathcal{B}$ never asked its own oracle to generate a signature for $(m_1, (\mathsf{pk}_1, \mathsf{time}_1))$, and can thus return all successful runs, and $(m_1, \sigma_1, \mathsf{time}_1, \mathsf{pk}_1)$, as for (m_1, time_1) is fresh by assumption, as $\mathcal{B}$ never queries its own oracle any longer for fresher time.

CA Unforgeability. Essentially the same reduction as for the weakly blind scheme.

CA Blindness. Let $\mathcal{A}$ be an adversary which breaks the CA blindness of our scheme. We can then construct an adversary which breaks the blindness of the used BSIG. $\mathcal{B}$ proceeds as follows. It generates pk_u honestly, which it also gives to $\mathcal{A}$, receiving $(\mathsf{pk}^*, \{m_0, m_1\}, \mathsf{time}, \mathsf{state}_1)$. It then gives state_1 to $\mathcal{A}$, and interacts with its own oracles like $\mathcal{A}$ does with his using m_0 and m_1, but uses $(\mathsf{pk}_u, \mathsf{time})$ as info. If $\mathcal{A}$ is finished it returns state_2, and $\mathcal{B}$ subsequently receives (σ_1, σ_2) from its own challenger. Then, $\mathcal{B}$ gives $\mathcal{A}$ state_2, and (σ_1, σ_2) to $\mathcal{A}$. Whatever $\mathcal{A}$ then outputs, is also output by $\mathcal{B}$. $\square$

Efficiency. We want to stress that in the first protocol message the user essentially proves knowledge of the secret key. If the signature on time is not valid, the protocol can directly be aborted. This prohibits that outsiders use the CA to check whether a given certificate is revoked. If this is not wanted for performance reasons, leaving this step out is also possible.

Clearly, both constructions require that a verifier needs to verify two signatures, while the CA has to generate a signature. However, considering that the CA has to vouch that a given certificate was not revoked, it has to generate a signature anyway, if the revocation information needs to be up-to-date, which clearly needs to be verified as well. In other words, our construction is already more efficient after the first signature verification. Moreover, compared to the approach by Boneh et al. [9], an outsider can trivially derive whether our protocol was used to generate the signature, which in turn increases trust in the signature itself, as the verifier can also decide whether it accepts a given pk_c as trustworthy.

4.1 Extensions

We now discuss informally how our basic constructions can be extended to account for additional use-cases. We omit full details and proofs due to space limitations, however the intuition should still become clear.

Signer-Anonymity. While both our constructions give message-privacy guarantees to the user, they reveal the identity of the signing party to the CA. If this poses a potential privacy problem, it can be mitigated as follows, for instance for the weakly-blind construction, cf. Fig. 9. The commitment is extended to also commit to pk_u. Then, instead of signing the tuple in Fig. 9 in the first step, one computes a signature proof of knowledge proving in zero-knowledge that one knows the secret key corresponding to the public key in the commitment, and that this public key is not on the blacklist. This can be done using similar techniques as Idemix [16].

Revocation into the Past. Our constructions are well-suited for situations where signing keys should simply be deactivated, e.g., when an employee leaves a company. However, in certain situations, it is also necessary to revoke "into the past" in order to also invalidate signatures issued between key leakage and revocation, cf. Fig. 1. In this case, the CA has to publish a list of revoked keys together with time-stamps of their revocation moment; upon verification, only signatures issued before this point in time would be accepted. From a complexity point of view this solution is similar to the combination of black-list based PKIs and time-stamping authorities, i.e., interaction is needed upon signing and verification.

Message Policies. One could also require that the signer proves (in zero-knowledge) that the message to be signed follows certain restrictions, e.g., that a company policy is followed. Only if the proof is valid, and the public key is not revoked, the server contributes to signature-generation. For example, a policy may be that a normal employee can only sign contracts below $1,000. This can even be done on a per-public-key basis. The size of signatures does not grow by this extension, and also the verification costs do not increase. Furthermore, the policy trivially remains hidden from the verifier.

Further extending the scheme efficiently such that also the CA does not learn any information about the policy remains a challenging open problem.

Robustness. Even though our security model is fixed for one signer and one CA, one can of course switch to a different CA on-the-fly. This protects against offline CAs, as one can simply use another one. In particular, a user can use a single signing key with different CAs, who act as revocation authorities for different domains (e.g., across different companies). Revocation by one CA does not affect other CAs. Security follows by a simple hybrid-argument.

Threshold Scheme. Related to the prior idea is an extension to threshold-cryptography. Namely, one could require that at least n-out-of-m servers need to participate in order to achieve robustness against offline servers.

5 Conclusion and Future Work

We have introduced the notion of CA-Assisted Signatures. These signatures enable the revocation of signing-rights if a secret is corrupted. This is achieved by letting a CA contribute to signature generation, vouching that the used public key was not revoked. Thus, signatures remain valid even after revocation of the certificate. Moreover, the CA can add timestamps, while neither the verifier nor the CA need to be online for verification. This has the additional benefit that verification requires less effort to check the validity of the signature. We furthermore propose various extensions increasing the privacy guarantees of our basic constructions.

Our construction does not pose any non-standard requirements to the signature scheme used by the user. In particular, existing signing infrastructures could thus easily be adapted to our design without the users having to change their key material.

References

1. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: CRYPTO. pp. 271–286 (2000)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: ASIACRYPT. pp. 452–473 (2003)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: ESORICS. pp. 159–177 (2005)
4. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. J. Cryptology 16(3), 185–215 (2003)
5. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: CCS. pp. 390–399 (2006)
6. Benaloh, J., Mare, M.D.: One-way accumulators: A decentralized alternative to digital signatures. In: Eurocrypt. pp. 274–285 (1993)
7. Bicakci, K., Baykal, N.: Server assisted signatures revisited. In: CT-RSA. pp. 143–156 (2004)
8. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: PKC. pp. 31–46 (2003)
9. Boneh, D., Ding, X., Tsudik, G., Wong, C.: A method for fast revocation of public key certificates and security capabilities. In: USENIX (2001)
10. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: EUROCRYPT. pp. 416–432 (2003)
11. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of Sanitizable Signatures Revisited. In: PKC. pp. 317–336 (2009)
12. Camenisch, J., Dubovitskaya, M., Enderlein, R.R., Lehmann, A., Neven, G., Paquin, C., Preiss, F.: Concepts and languages for privacy-preserving attribute-based authentication. J. Inf. Sec. Appl. 19(1), 25–44 (2014)
13. Camenisch, J., van Herreweghen, E.: Design and implementation of the *idemix* anonymous credential system. In: CCS. pp. 21–30 (2002)
14. Camenisch, J., Koprowski, M., Warinschi, B.: Efficient blind signatures without random oracles. In: SCN. pp. 134–148 (2004)
15. Camenisch, J., Lehmann, A., Neven, G., Samelin, K.: Virtual smart cards: How to sign with a password and a server. ePrint 2015, 1101 (2015)
16. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: CRYPTO. pp. 61–76 (2002)
17. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO. pp. 199–203 (1982)
18. Chow, S.S.M., Hui, L.C.K., Yiu, S., Chow, K.P.: Two improved partially blind signature schemes from bilinear pairings. In: ACISP. pp. 316–328 (2005)

19. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (2008)
20. Derler, D., Hanser, C., Slamanig, D.: Revisiting cryptographic accumulators, additional properties and relations to other primitives. In: CT-RSA. pp. 127–144 (2015)
21. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: CRYPTO. pp. 307–315 (1989)
22. Fischlin, M., Schröder, D.: Security of blind signatures under aborts. In: PKC. pp. 297–316 (2009)
23. Fischlin, M., Schröder, D.: On the impossibility of three-move blind signature schemes. In: EUROCRYPT. pp. 197–215 (2010)
24. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Comp. 17, 281–308 (1988)
25. Gutmann, P.: PKI: it's not dead, just resting. IEEE Computer 35(8), 41–49 (2002)
26. Huang, X., Susilo, W., Mu, Y., Zhang, F.: On the security of certificateless signature schemes from asiacrypt 2003. In: CANS. pp. 13–25 (2005)
27. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures (extended abstract). In: CRYPTO. pp. 150–164 (1997)
28. Krenn, S., Samelin, K., Sommer, D.: Stronger security for sanitizable signatures. In: DPM. pp. 100–117 (2015)
29. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: CCS '96. pp. 48–57 (1996)
30. Milles, D.L.: Time synchronization in DCNET hosts. Tech. rep., COMSAT Laboratories (1981)
31. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: TCC. pp. 80–99 (2006)
32. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO. pp. 129–140 (1991)
33. Pöhls, H.C., Samelin, K.: On updatable redactable signatures. In: ACNS. LNCS, vol. 8479, pp. 457–475. Springer (2014)
34. Rivest, R.L.: Can we eliminate certificate revocations lists? In: FC. pp. 178–183 (1998)