Improved Privacy for Two Cloud Scenarios Using Advanced Signature Schemes

Karl Koch and Roland Urbano

IAIK, Graz University of Technology, Graz, Austria
{karl.koch,urbano}@student.tugraz.at

Abstract. Nowadays cloud-based services are practically ubiquitous and many enterprises follow the trend to migrate (parts of) their infrastructures and services to cloud service providers (CSPs). However, CSPs can often not be considered as being fully trusted. Yet, one can often benefit from employing cryptographic means to (1) enforce the honest behaviour of the CSP and (2) prevent the CSP from learning privacy sensitive information.

In this paper, we describe two potential real-life cloud service scenarios from a rather sensitive field (eHealth) and a very common field (identity-provisioning/single-sign-on) and infer accompanying authenticity and privacy requirements. Based on these requirements, we then analyse how the properties provided by different variants of advanced signature schemes can help to cover (subsets of) those requirements. In particular, we try to map the requirements to the respective properties provided by the signature primitives and investigate their benefits over naive realizations of the envisioned scenarios. Most notably, we can show that rather unknown but entirely practical advanced signature primitives provide significant benefits over naive realizations of the scenarios.

1 Introduction

In the last few years we have seen ubiquitous cloud computing growing extremely fast and more and more devices are connected while sharing a vast amount of data. Recently, the common understanding to protection of data privacy changed tremendously [6] due to different privacy-related incidents [12] all around the world and leaks about large scale surveillance efforts by intelligence agencies [5]. Since privacy is a fundamental and important right, our aim is to contribute to enhance users' privacy and security in the realm of cloud-based services.

In this work we will investigate two real-life scenarios, one from the eHealth domain and one from the field of identity-provisioning/single-sign-on. Triggered by recent security incidents, we investigate desired privacy and security properties and study how these properties can be accomplished by means of (1) conventional signature schemes and (2) three advanced signature primitives. Lastly, for each scenario we will also discuss the benefits of a realization with modern schemes over a realization with a conventional scheme.

The first eHealth scenario considers a cloud service hosting medical data from different caregivers and enabling a patient to access her medical records online[1] as well as to share her records with other entities like her employer or insurance agent. Common privacy problems with this scenario are the unwanted propagation of sensitive medical information about a patient or the unwanted disclosure of specific information to some party, e.g., an employee may not want to reveal the exact diagnosis to the employer when providing a sick note.

In the second scenario, we consider some cloud service providing multiple applications and a set of identity providers. Our main focus will be on the authentication process and the importance of the "one account for everything" principle. Practical privacy problems with this scenario are for instance that users from one specific identity provider are identifiable by all services due to the token they get from this identity provider to attest their identity towards the service. Another problem is the reuse of a stolen or leaked token. Microsoft, for instance, quite recently experienced an attack, where an attacker stole a login token from a user via a malicious URL and got access to all services provided to this user by only having a single token [13].

1.1 Advanced Signature Primitives

Subsequently, we discuss some signature schemes which provide interesting properties in our context and informally revisit their security properties. All those schemes are required to be correct, which means that everything works correctly if everyone behaves honestly. For the sake of compactness we do not explicitly mention correctness subsequently and implicitly assume the correctness requirement for every scheme.

Redactable Signatures. Using conventional digital signatures, every alteration of a signed document invalidates the signature. In contrast, redactable signatures (RSs) [8,11] allow the signer to define admissible redactions of the signed message so that everyone can later black out the admissibly redactable parts without invalidating the signature.

An RS is required to be *unforgeable* and *private*. Informally those properties are defined as follows. Unforgeability says that the only possibility to come up with a valid redacted signature is to start from a valid signature from the original signer. Finally, privacy requires that it is infeasible to reconstruct the original message from a redacted version of this message.

Group Signatures. In group signature (GS) schemes, which were initially envisioned by Chaum and van Heyst [2], a group manager sets up a group and every member of the group is later able to create signatures on behalf of the group. Thereby, a group signature does not reveal the identity of the actual signer and can be publicly verified using the group manager's public key. In addition, there is an authority called opening authority, which—e.g. in case of a dispute—can identify the actual signer when given a valid group signature on some message.

[1] Such types of records are often called personal health records (PHRs).

For our analysis, we build upon the formalization of static GSs by Bellare et al. [1]. In their model, group signatures are required to provide *traceability* and *anonymity*. We informally revisit those properties below. Traceability requires that every valid group signature can be traced back to the actual signer of the message. Anonymity requires the infeasibility to determine the identity of the actual signer of a group signature for everyone except the opening authority.

Besides that, GSs may provide means for user revocation, i.e., means to exclude group members from the group.

Designated Verifier Signatures. Designated verifier signatures (DVSs), firstly introduced in [7] and refined in [9], allow to create signatures which can only convince a designated verifier of the authenticity of the signature.

DVSs are required to be *unforgeable* and *non-transferable*. Informally *unforgeability* requires that the original signer and the designated verifier are the only entities who can come up with respective valid signatures. *Non-transferability* captures the requirement that—without the designated verifier's secret key—one cannot distinguish honest signatures of the signer from simulated signatures of the designated verifiers.

An extension of DVSs, which is also useful in our context, are universal designated verifier signatures (UDVSs) [10]. UDVSs generalize the concept of DVSs in the sense that the signer issues a publicly verifiable signature which can then be turned into a UDVS by a third party.

2 Scenarios

Followingly, we further elaborate on two real-world scenarios, outlining privacy requirements in cloud-based realizations of different systems and discuss how these requirements can be met first, by using a Conventional Signature Scheme (CSS), and second, by using the primitives introduced in section 1.1 (Advanced Signature Schemes (AdSSs)). Finally, we will compare the two realizations for each scenario.

All requirements are specified with respect to the honest behavior of all included parties. Moreover, we require that transmissions between entities are secured by state-of-the-art cryptography.

2.1 eHealth

Nowadays digital recording of health-related data, generally known as eHealth, is getting more and more popular. Subsequently, we discuss a potential eHealth scenario, identify its requirements, and then show how the identified requirements can be met by using a CSS. Further on, we set the requirements into the context of the properties provided by the primitives discussed in the introduction. Finally, we will compare the two solutions.

Imagine a patient who goes to a hospital and is checked by a doctor. The doctor compiles a report, signs it, and this signed report gets stored in some sort of cloud storage, so that the patient can easily access it. Then the patient may want to share only different parts of the report with different stakeholders such as health insurances, medical research labs, or other doctors, because these entities have different interests. Henceforth, this "blacking out" of certain parts in an authentic report is called redaction.

Figure 1 illustrates our envisioned scenario. The patient provides the hospital's report to the target person(s) after three phases: (1) the patient gets checked by the hospital; (2) the signed report gets stored in the cloud; (3) the patient may (individually) redacts parts of the report; (4) the respective target person can access the (redacted version of the) signed report.

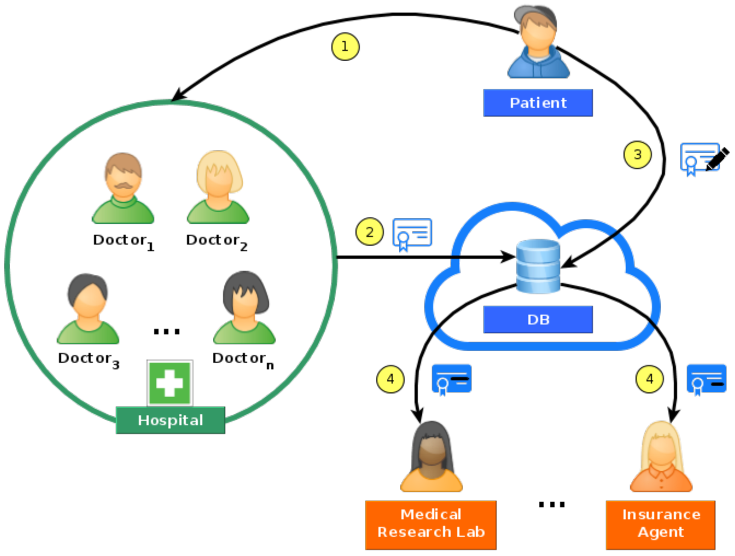

Fig. 1. The procedure in our eHealth scenario.

We now continue with the identification of requirements in our eHealth scenario:

Think of a doctor who signs a health report for some patient using a CSS. Clearly, every verifier of such a report can already derive sensitive information from the identity of the doctor—potentially even the disease of the patient. Thus it is desirable that the verifier is only convinced of the report's authenticity with respect to all doctors in the hospital, while the identity of the signing doctor remains concealed.

Requirement 1: *Signatures hide the identity of the signer among the group of all potential signers.*

Though in some cases, e.g., in case of a wrong diagnosis or malpractice, it is required that the hospital can determine the signer of a document. Thus, there must be an additional entity, who can reveal the signer's identity.

Requirement 2: *There must be an entity, who—given a valid signature on some report—is capable of unambiguously identifying the original signer.*

Furthermore, if a doctor leaves the hospital, it should no longer be possible for her to create valid signatures.

Requirement 3: *It must be possible to revoke the doctor's signing rights.*

Besides that, the patient may want to remove certain parts of the signed report for privacy reasons while authenticity is still ensured. For instance personal data might be redacted when handing on reports to a medical research lab.

Requirement 4: *Documents, signed by some doctor, must be redactable without invalidating the signature.*

Moreover is it required that forbidden redactions of a document invalidate the signature to prevent unauthorized modifications. Otherwise a patient could, for example, extend the duration of a sick leave from 1 week to 1 month.

Requirement 5: *Forbidden redactions of signed documents must invalidate the signature.*

Additionally, the patient may want to convince only the intended receiver, i.e., the designated verifier, of the authenticity of a certain report, when sharing the (redacted version of the) signed report. So that an outsider can never be sure whether a given document is authentic in case of data leakage or when the intended receiver further forwards the report.

Requirement 6: *Patients must have the possibility to define the intended receiver of a document so that authenticity of the document can only be verified by the respective receiver.*

Realization with a CSS. We now discuss a potential solution for the aforementioned requirements by using a CSS. In particular, we use only the conventional signing and verification operation with the corresponding key pair. This means that, when someone modifies the contents of a report, the signature of the report is not valid anymore. Moreover, everyone who is in possession of the verification key can verify the authenticity of a report, while only someone who is in possession of the signing key can create valid signatures on reports. To achieve a convenient way of signing and sharing health-related data, we envision (as stated in the description of our eHealth scenario) an easy-to-use cloud-based service.

The main functionalities of the cloud service are (1) the uploading of reports for doctors; and (2) the downloading/sharing of (redacted) authentic reports for patients. Every doctor and patient of the hospital can access these functionalities by means of access control.

Signing of reports works as follows: (1) doctors upload the report; and (2) if a patient downloads/shares a (redacted) report, it gets signed by the cloud service. In this way, even when doctors have access to the system, they have no access to the private signing key. Furthermore does the cloud service use only a single signing key. Therefore, given only a signed report, no one can identify the corresponding doctor.

In order to accomplish re-identification of responsible doctors, the cloud service could store the information about the doctors and the corresponding reports on some list. This, however, would require rather strong trust assumptions.

If a doctor leaves the hospital, her access to the system will be revoked. Thereby, only active doctors of the hospital are able to upload reports.

When a patient wants to share only certain parts of a signed report with some third party, he needs to send the respective request to the cloud service. When this third party then wants to access the resulting report, the cloud signs the redacted version of the report and sends it to the party.

One open issue is, that a person who is in possession of a signed report, could forward the authentic report and information about the verification key to some other third party. Since this is in the nature of a conventional signature, it is not possible for a patient to convince only the intended receiver of the authenticity of a report.

Realization with AdSSs. Now we discuss a potential realization using AdSSs. Again, as for the realization with the CSS, we envision an easy-to-use cloud service which provides our entities a convenient way of signing and sharing health-related data. Followingly, we will describe how the proposed solution tries to compose the signature primitives introduced in 1.1, and thereby meet our entities needs.

Firstly, we can employ GSs so that doctors can sign documents for patients while staying anonymous with respect to verifiers. Then, the opening authority can still identify the original signer in case of a dispute. Moreover, if a doctor leaves the hospital, by using means for revocation one can revoke a doctor's signing key.

Furthermore, by composing GSs with RSs we achieve the redactability of signed health records. Derler et al. proposed a generic RS scheme construction from CSS and cryptographic accumulators [4]. Thereby, the accumulator is used as a mean to encode the message, while the CSS is used to sign this encoding. We observe that this paradigm allows to replace the CSS by a GS while maintaining the expected security properties (under a slightly adapted unforgeability definition taking a multi-user setting into account). Also the other desired properties of GSs, as mentioned before, and of RSs, the possibility to redact authentic reports, is obtained within such a combination. Patients can redact signed reports by using the corresponding feature in the cloud service, even without a doctor's private signing key. Now, every time a user wants to share an authenticated (redacted) report to some entity, the authenticity of such a document would be publicly verifiable by everyone.

Since everybody is able to verify the authenticity of a (redacted) report, one could ask whether it is possible to additionally employ UDVSs. So that patients could create a designated-verifier version of the original signature via our cloud service every time they forward a report to a third party. Then, some other third party could not distinguish if the signature was created by the patient or the designated verifier.

The combination of GSs, RSs, and UDVSs is, however, not as intuitive as the combination of GSs and RSs. That is why we do not further investigate the combination of all introduced cryptographic primitives (1.1) in this scenario. Though, it would be very interesting to see further research in that area. As for example Derler et al. [3] formally modeled such a combination.

Comparison between CSS and AdSSs. Subsequently, we will compare the realization with a CSS and AdSSs. With our comparison we want to analyze if AdSSs outweigh the CSS. We analyze by looking at each requirement, and finally conclude.

Anonymity, traceability, and revocation can be met with both schemes. However, in the first solution one needs rather strong trust assumptions in the cloud service, whereas the properties are cryptographically ensured in the second solution.

Redaction of authentic reports can be obtained with both realizations as well. Again, to achieve the desired privacy feature with the CSS, one needs to completely trust the cloud service regarding the redactions. On the contrary to the second approach, one can cryptographically ensure correct modifications of reports with RSs by defining admissible redactions. Moreover is it for both realizations infeasible to forge valid signatures, if one does not have the correct signing key.

For patients the possibility to define an intended receiver when sharing a signed report seems to be rather difficult as we do not see a straight forward way to combine GSs, RSs, and DVSs. While with the CSS non-transferability is not possible at all, the realization with AdSSs seems to offer a potential solution (but as already mentioned, one such direction is discussed in [3]).

Table 1 gives an overview of which requirements are covered by our realizations (assuming that such a straight forward composition is possible at all).

Due to the aforementioned facts, our realization with AdSSs clearly outweighs the realization with a CSS.

2.2 Authentication for Cloud Services

For the second scenario we discuss a potential real world cloud scenario with multiple services hosted by a Cloud Service Provider (CSP). A CSP is a company offering services like document editing, streaming or storage online to the user via different websites or mobile apps. Big CSPs mostly offer not just a single service

	RS		GS			DVS
	Priv.	Uf.	Anon.	Trac.	Revo.	Non-Tran.
Rq. 1:			✓			
Rq. 2:				✓		
Rq. 3:					✓	
Rq. 4:	✓					
Rq. 5:		✓				
Rq. 6:						✓ (X)

Table 1. Mapping from the requirements to the respective property by our realizations in the eHealth scenario. We use the following acronyms for the respective properties: Privacy (Priv.), Unforgeability (Uf.), Anonymity (Anon.), Traceability (Trac.), Revocation (Revo.), and Non-Transferability (Non-Tran.). Legend: ✓ . . . supported, X . . . not supported. If the realization with CSS differs from the one with AdSSs, we put the mapping from the CSS into brackets.

online, but rather a whole set of personalized services. To increase usability, CSPs offer single sign-on to allow usage of all services with just one account without re-authentication to each service. Alternatively they also allow users to authenticate with different identity providers and then use the received Authentication Token (AT) across all cloud services to attest their identity. These identity providers don't have to be part of the current CSP, but can also be different CSPs or companies.

In our scenario we will focus on the connection between the used cloud service and the identity provider issuing the corresponding AT, which is used to authenticate with the CSP. This offers interesting possibilities to provide enhanced privacy features for both, the issuer and the verifier. The overall setup, as illustrated in Figure 2, consists of several cloud services offered by a CSP and different identity providers. The CSP specifies a set of identity providers which are trusted to issue ATs. On login to the CSP's cloud the user requests an AT from one identity provider where he is registered, and then uses the returned AT to authenticate with the targeted cloud service: The used AT is checked by the CSP and the access to the cloud is denied if validation fails. Additionally we consider an adversary controlling one of the services who can access all AT on this service.

We further assume that the traffic between identity provider, user and CSP is encrypted and somehow hidden through an anonymous overlay network such as Tor[2] or similar. Hence, intercepting the traffic to spy on transmitted messages is not easily possible and out of the scope of our threat model.

We continue with an informal listing of desired requirements for the cloud service scenario. These requirements aim to enhance the security and the privacy properties for the services as well as for the identity provider by trying to remove

[2] https://www.torproject.org/

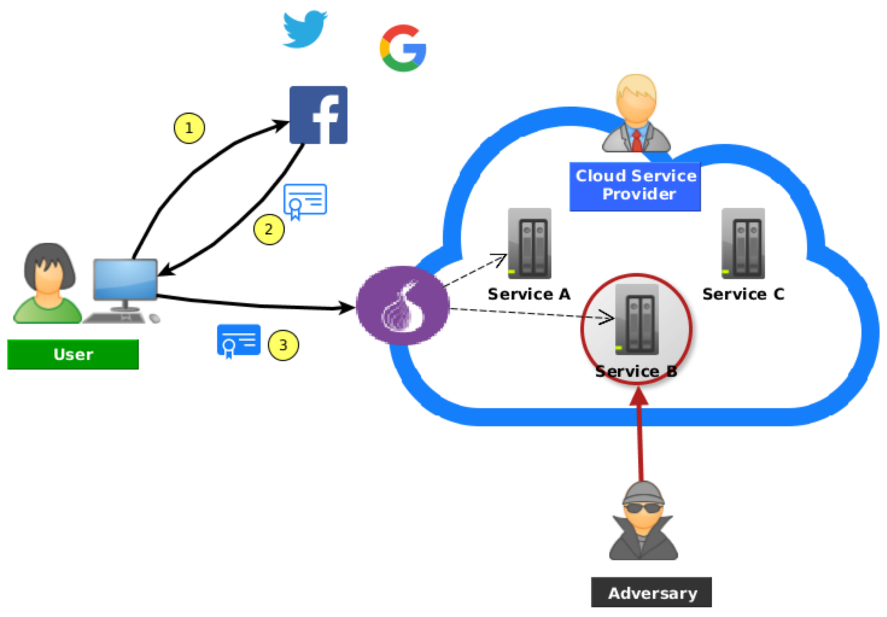

Fig. 2. The procedure in our CSP scenario.

the direct link between them.

Firstly, it is desirable that only those identity provider can issue valid ATs, which are given the permission to generate ATs by the respective CSP.

Requirement 1: *Only authorized identity providers are able to create valid ATs.*

Further ATs must only work for one single service and only the receiving service should be in possession of the required knowledge to verify the validity. Any other service, given the AT, must not be convincible of the validity of the AT. As already mentioned, recently Microsoft experienced such an attack where a leaked token was used to access all of the user's personalized cloud services. [13]

Requirement 2: *ATs must only be usable/valid/verifiable for the receiving cloud service.*

We also require that the AT can somehow prove the authenticity of the user to the targeted service, but to the targeted service only.

Requirement 3: *Validity of ATs can only be verified by the targeted service.*

Authentication with a selected identity provider requires prevention against linkage of AT to the issuing identity provider. In particular the receiver of a signed AT (or an adversary) must not learn which identity provider issued the AT.

Requirement 4: *The identity of the issuing identity provider must not be revealed by the AT.*

Lastly, there might be some cases where it would be good to know who exactly issued a questionable AT. At least a neutral entity (or the CSP) should be able to "unmask" the original identity provider which issued the AT. This should ensure honest behavior of all issuing identity providers.

Requirement 5: *A dedicated entity must be able to trace the issuer's identity when given an AT.*

Realization with a CSS. The construction/implementation, in general, should yield the non-transferability of the AT to other services and the anonymity of the AT's issuing identity provider. Furthermore we define the threat model for this scenario to only respect adversaries which are outside the cloud but have access to leaked information such as recently used AT. Due to this, we declare the CSP including all services to be trusted in respect to the handling of AT.

Since non-transferability of an AT in this scenario requires that an adversary cannot forward a leaked AT to a different cloud service, which is also able to verify the signature, we propose that users create key pairs for all receiving services. So for service A, the user would use his signing key sk_A and for service B sk_B. Each service only accepts if the signature is valid with respect to the correct key. While this solves the first issue, it prevents to reach anonymity of the targeted service against an adversary, as all verification keys are public and she can determine the receiving service by trying all keys for verification.

Anonymity of the identity provider is also possible, but difficult to achieve since it requires indistinguishability between different signing authorities. The simplest approach would be to share one signature key pair among all signing parties (identity providers). Yet, this would require a manual setup and key distribution as well as a new key generation each time a member of the signing group leaves or a new enters. Additionally, the identity of a signer in question cannot be revealed easily afterwards, without keeping track of mappings between identity provider and issued AT. A possible realization of this mapping would be a database or a list maintained by a neutral entity, who would act as the trusted manager of this list and could reveal specific identities if needed.

Realization with AdSSs. Followingly we will propose a potential system where we try to combine two of the introduced signature primitives to solve all the identified requirements.

For each service, the user uses a UDVS on the AT to ensure that only the targeted service is convinced of the authenticity of the new AT'. In particular the targeted service is the designated verifier, who is the only receiver able to verify the authenticity of the AT'. This ensures that nobody else can reuse a leaked AT from one service for another one. UDVSs are also unforgeable for an adversary. If an AT' is leaked from one service and the adversary forwards it to another service, to impersonate the original owner of the AT', the new receiving service will not be able to determine if the signature was created by some user or "faked" by the primary receiver.

Furthermore it seems that GSs are useful to provide identity provider anonymity, although it is not quite clear how to combine UDVSs with GSs to preserve the security properties of both. Therefore we leave this as an open issue, since it would require a formal model of this scenario.

Comparison between CSS and AdSSs. Similar to the first scenario we followingly compare the described solutions from above in detail and point out the advantages of AdSSs over a CSS.

Starting with the CSP's cloud services we argue that, using a UDVS strongly enhances the privacy features for the scenario. With a CSS the user needs as many key pairs as services exist, and the services need additional user-specific knowledge. UDVSs therefore outperform CSSs by requiring only one user-specific key pair, but still are able to designate the signature for one receiver only. In addition UDVSs do not require the same user-specific knowledge as a CSS realization.

The required identity provider anonymity and also the traceability are achievable with both realizations. However, with a CSS there are some limitations we have to consider. Although anonymity could be achievable with a shared key pair, it is very unlikely to get multiple identity provider to share a "secret" key, and because it is such a rather strong trust assumptions, we argue that it is impractical to use in a real scenario. In contrast, the realization with AdSSs could use a GS to hide the identity provider and provide traceability as well. And if one identity provider is removed from the group, with a GS the group manager can revoke the corresponding key for this specific identity provider without need for a new group key to be distributed.

Table 2 highlights the mapping of our requirements to the properties of the respective signature schemes. Only simultaneously covered requirements are considered to fulfill the corresponding properties.

All in all, our proposed combination provides more privacy features and are much more comfortable to use. Opposite to CSSs, which, following from above, need a lot more work to be done to achieve parts of the desired requirements. We therefore believe, it pays off to put effort into formally model the scenario and find a practical realization.

3 Conclusion

In this paper we introduced three AdSSs and showed how these could be combined to enhance the privacy properties of two possible cloud scenarios.

	DVS			GS	
	Uf.	Auth.	Non-Tran.	Anon.	Trace.
Rq. 1:	✓				
Rq. 2:			✓		
Rq. 3:		✓			
Rq. 4:				✓(X)	
Rq. 5:					✓(X)

Table 2. Mapping from the requirements to the respective property by our realizations in the cloud service scenario. We use the following acronyms for the respective properties: Unforgeability (Uf.), Authenticity (Auth.), Non-Transferability (Non-Tran.), Anonymity (Anon.), and Traceability (Trac.). Legend: ✓ . . . supported, X . . . not supported. If the realization with CSS differs from the one with AdSSs, we put the mapping from the CSS into brackets.

In the first scenario we described a possible solution for an eHealth cloud service using a combination of GS, RS and UDVS. We showed that the combination of GS and RS is straight forward, but the addition with UDVS isn't.

In the second scenario it was all about Cloud Service Providers and the privacy enhancement using GS and UDVS. As proposed in the first scenario, we argued why the combination of these primitives seems to hold the expected properties, but since we could not identify a simple interface for combination we leave this part for future research.

Further we believe, it could be of interest to formally model the second scenario to prove the combination of GS with UDVS in respect to the requirements formulated in this work. In the meantime we are working on a practical implementation of the first scenario.

Acknowledgements. We thank David Derler, Stephan Krenn, and Daniel Slamanig for suggesting the first use case and the selection of the cryptographic primitives, as well as providing us with their manuscript of the formal model for the combination of group signatures, redactable signatures, and designated verifier signatures.

References

1. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Advances in Cryptology - EUROCRYPT 2003. pp. 614–629 (2003)
2. Chaum, D., Van Heyst, E.: Group signatures. In: Advances in Cryptology-EUROCRYPT'91. pp. 257–265. Springer (1991)
3. Derler, D., Krenn, S., Slamanig, D.: Signer-anonymous designated-verifier redactable signatures for cloud-based data sharing (2016), manuscript
4. Derler, D., Phls, H.C., Samelin, K., Slamanig, D.: A general framework for redactable signatures and new constructions. In: International Conference on Information Security and Cryptology. pp. 3–19. Springer (2015)

5. Glen, G.: Nsa prism program taps in to user data of apple, google and others. https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data (2013), accessed: 2016-07-22
6. Gorodyansky, D.: Privacy and security in the internet age. http://www.wired.com/insights/2015/01/privacy-and-security-in-the-internet-age/ (2015), accessed: 2016-07-22
7. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Advances in Cryptology-EUROCRYPT'96. pp. 143–154. Springer (1996)
8. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Topics in Cryptology, CT-RSA 2002, pp. 244–262. Springer (2002)
9. Lipmaa, H., Wang, G., Bao, F.: Designated verifier signature schemes: Attacks, new security notions and a new construction. In: Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Proceedings. pp. 459–471 (2005)
10. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Advances in Cryptology - ASIACRYPT 2003. pp. 523–542 (2003)
11. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Information Security and Cryptology-ICISC 2001, pp. 285–304. Springer (2001)
12. Thielman, S.: Hackers release new ashley madison data targeting site's ceo and operators. https://www.theguardian.com/technology/2015/aug/20/hackers-new-ashley-madison-data (2015), accessed: 2016-07-22
13. Wei, W.: Microsoft pays $13.000 to hacker for finding authentication flaw. http://thehackernews.com/2016/04/microsoft-bug-bounty.html (2016), accessed: 2016-04-20