Generalizations and Extensions of Redactable Signatures with Applications to Electronic Healthcare

Daniel Slamanig[1] and Stefan Rass[2]

[1] Carinthia University of Applied Sciences, 9020 Klagenfurt Austria
d.slamanig@cuas.at
[2] Klagenfurt University, System Security Group, 9020 Klagenfurt Austria
stefan.rass@syssec.at

Abstract. Redactable signatures allow for altering signed documents, retaining the validity of the signature without interaction with the original signer. In their plain form, such schemes are designed for documents having an unspecific structure, i.e. documents are simply considered as binary strings. In this work, we generalize the concept of redactable signatures towards documents that inherently provide a structure and investigate the security of our construction. Furthermore, we present extensions to our scheme, adding features not commonly provided by other redactable signature schemes. Additionally, various applications in healthcare are discussed, supporting the applicability and usability of our construction.

1 Introduction

A standard digital signature links an identity to a document and thereby provides an authentication mechanism, as well as integrity assurance. Consequently, no alteration of a signed document is possible once the signature has been attached. However, there are scenarios where it would be valuable to have the possibility to replace or remove (specified) parts of a message after signature creation such that the original signature stays valid. Such signatures are called *redactable* (we remark that the naming is not consistent throughout the literature). An example application is the anonymization (removal of identifying information) of medical documents or general XML-documents.

Redactable signature schemes were independently introduced by *Steinfeld et al.* [1] and *Johnson et al.* [2] in 2001 and 2002 respectively. Loosely spoken, such signature schemes are constructed from standard signature schemes using the hash-then-sign paradigm by virtue of modifying the construction of the hash value. Merkle-trees [3] are a convenient tool for that matter: if a message is hashed via a Merkle-tree, then a message part at a leaf of that tree can be redacted by replacing it with an empty-symbol in the message, along with the hash-value of the (now missing) message part. For signature verification, the remaining hash-tree can be constructed as usual, thus creating a signature with

redaction capabilities. One such scheme is described in section 4.
Contribution: The contributions of this paper are manifold: while standard redactable signatures are applicable to documents providing no specific structure, i.e. which are simply interpreted as binary strings, and organized as leafs of binary trees, we generalize the whole concept towards documents providing an inherent structure. We present efficient constructions with provable security and present various extensions, adding valuable features to the given construction.

2 Applications in Electronic Healthcare

Although there are several different applications of signature schemes supporting redaction, we will focus on their application in electronic healthcare (eHealth) here and thus briefly motivate the field and their potential benefits.

In recent years eHealth and ubiquitous healthcare have gained significant attention in the scientific community as well as among practitioners, since these concepts hold great potential to improve medical treatment processes. In order to guarantee an adequate quality of data, it is important that authenticity and integrity of involved data can be checked at any time. Hence, digital signatures seem to be an invaluable tool for that matter. Redacting parts of documents after signing while preserving the signature validity can be valuable, when information of a medical document needs to be removed (e.g. the document is anonymized). This spares the need to contact the original signer to re-create the signature and enables more time-efficient medical processes with guaranteed signature validity. More importantly, *source-authenticity* of the original data source is ensured.
Achieving k-Anonymity: Person related health data are in general very sensitive information and consequently must be protected appropriately. Especially, when using these data for secondary use, which includes medical research, clinical studies and second opinions, the protection of the privacy of patients is obligatory. Hence, it is necessary to prepare data such that the patients cannot be identified uniquely anymore, before passing it to another party for secondary use. This process of preparation focuses primarily on attributes that directly identify the patients and secondarily on attributes that indirectly identify the patients. Redactable signatures allow for removing certain identifying attributes in order to anonymize a document whilst retaining its authenticity. Suppose that a combination of non-identifying attributes uniquely identifies an individual (a study [4] from the year 2000 estimated that 63% of the population of the United States can be uniquely identified based only on gender, date of birth, and 5-digit zip code). Assume that upon removing a subset of those attributes (*suppression*), identification becomes ambiguous, meaning that these attributes can equally well be related to k different individuals. The same effect could be achieved by replacing the attribute values by more general ones (such as telling only the first 2 digits of a ZIP-code for instance) This process is called *generalization* (see [5] for details). Either way, we get *k-anonymity* of the individual, as introduced in [6, 7]. Removing or replacing attributes to achieve k-anonymity is

a natural application of redactable signatures. We revisit this in section 6.
Privacy Preserving and Unbiased Second Opinions: Redacting certain parts of an authentic (signed) medical document may as well be desirable if second opinions for some medical problem are sought. Either a medical expert or the patient could call for anonymity of the patient (privacy), or could redact certain parts of the medical document to make sure that the other expert remains unbiased by otherwise visible information, e.g. the diagnosis, while having authentic access to relevant information, e.g. anamnesis, lab-results, etc.

3 Related Work

Signature Schemes: Sanitizable signatures [8, 9] permit a designated party (*sanitizer*) to remove or replace desginated parts of a document. The signer can exactly mark parts as (not) redactable. However, the signer cannot specify how this redaction may look like, i.e. the redactor can replace a redactable part with an arbitrary string. The redaction capability is achieved by computing chameleon hashes (trapdoor commitments) for all the parts of a message that can be redacted. Only the sanitizer, in possession of the trapdoor information, is able to compute collisions for the chameleon hashes and hence can change the respective parts of the message arbitrarily. Nevertheless, for a verifier it is not noticeable whether the original message has been modified. The concept of sanitizable signatures was further extended to trapdoor sanitizable signatures in [10], which allow to give the power of sanitization to possibly several entities. Recently, there have also been proposed some extensions that restrict the choice of replaced strings [11], which seems to be very important and is also discussed in this paper in the context of redactable signatures.

In content extraction signatures [1], which, besides other constructions, use a hash-tree based scheme similar to redactable signatures, everybody can remove substrings. Furthermore, the parts that can be removed can be specified by means of a policy named content extraction access structure. This policy, which specifies the set of indices of redactable document parts, is signed along with the message and needs to be evaluated by the signature verification algorithm. If any redaction has taken place that is not specified in the policy, the signature validation algorithm rejects the signature.

A straightforward approach to redactable signatures is by splitting the document into pieces, each of which is signed separately. This makes removing a part simple but computationally expensive and not very flexible. An efficient construction is possible by using the pairing-based signature of [12], which saves bandwidth by aggregating all single signatures into one signature. However, all signatures corresponding to message parts that can be redacted need to be available to the redactor in addition. Using this construction, it is possible to remove blocks from the document and to remove the corresponding signatures from the aggregated signature without the verifier being able to realize whether the message has been redacted or not (invisible redaction) [13]. This concept was recently extended to also provide non-invisible redaction [14], which can also

be achieved for the aforementioned construction by including indexes into the single message parts. Another scheme, based on aggregating single signatures using batch RSA signatures, was introduced in [1] (prior to the aforementioned schemes). Although, their scheme also requires a number of RSA signatures that is linear in the number of message blocks, once a redaction has taken place, solely a single RSA signature (as the product of single signatures) needs to be transmitted to the verifier.

Furthermore, [15] have recently extended the approach of [2], such that a signer can specify which parts of a document are allowed to be redacted. Their approach is similar to the one of [1]. Another approach by [16] extends the tree-based construction of [2] such that the length of the redacted strings can be hidden. Finally, [17] have proposed a redactable signature scheme that does not rely on random oracles and is provably secure in the standard model.

Medical Applications: *Ateniese et al.* [9] mention alongside that sanitizable signatures can accommodate different level of data de-identification, supporting the "minimum necessary" disclosure standard of HIPAA Privacy Rule, but do not consider further details. The authors of [18] have recently proposed a concept to apply redactable signatures in context of Personal Health Records (PHRs) to realize minimum disclosure of personal information and to achieve cryptographic enforcement of dependencies between parts containing personal information. Another recent work developed independently from ours by *Haber et al.* [15] can be considered as closest to the ideas presented in this paper. They present redactable signatures that can be used for pseudonymization and de-identification of data, although they mainly focus on relational, i.e. 2-dimensional, data tables. In contrast, our work focuses on applying the aforementioned and additional mechanisms on structured documents.

4 Redactable Signatures

As mentioned in the introduction, redactable signatures permit removing parts of a signed document whilst retaining the validity of the signature. Technically, this amounts to an efficient update of the signature that requires no interaction with the original signer. Our proposed scheme will build upon the scheme of *Johnson et al.* [2], which we will briefly describe. We assume the reader to be familiar with the concept of a (binary) Merkle-tree, as this is used extensively in the following. The second core ingredient for our construction are pseudorandom generators (PRGs) based on GGM-trees [19].

Johnson-Molnar-Song-Wagner Redactable Signature Scheme: To construct a redactable signature for a document D, the authors of [2] divide the document into $N = \left\lceil \frac{|D|}{n} \right\rceil$ blocks of size n, so that $D = D[1], \ldots, D[N]$. Next, they build a complete binary tree with N leafs, having the message blocks assigned to them in the correct order. Consequently, one is able to build the hash of the root node h_R of the tree by hashing up the tree from the bottom (i.e. simply construct a Merkle-tree) and sign h_R by using a conventional signature scheme, i.e. set $\sigma = S_{SK}(h_R)$.

To remove the i-th block, which can be seen as a replacement with the empty symbol $\perp$, one may pass the document $D' = D[1], \ldots, D[i-1], \perp, D[i+1], \ldots, D[N]$ to a verifier, along with the hash value $h_{D[i]} = H(D[i])$ of the removed message block (leaf) $D[i]$. The missing message block causes no problem in reconstructing the Merkle-tree, because its hash-value is still available. The updated signature is simply $\sigma' = (\sigma, h_{D[i]})$, where σ is the original signature. One problem instantly arises if the block size is too small to prevent brute-force determination of the missing part $D[i]$. Therefore, a GGM-tree with its shape identical to the Merkle-tree is used to generate randomizers that are attached to each intermediate value (or message block) prior to hashing it. Technically, one generates a randomizer r_{v_i} for every node v_i of the tree by means of a GGM-tree, starting with a randomly chosen seed r_{v_R} at the root node v_R. Instead of solely hashing the values (message parts denoted as x), as within the Merkle-tree, the subsequent rules (1) are applied.

$$\phi(p) = \begin{cases} H(\phi(c_l)||\phi(c_r)) & \text{if } p \text{ has two children,} \\ H(\phi(c_l)) & \text{if } p \text{ has one child,} \\ H(r_p||x) & \text{if } p \text{ is a leaf.} \end{cases} \tag{1}$$

Intuitively, the values of the leafs are hashed along with a randomizer, which allows to hide the values of removed parts of the document. Technically, this can also be seen as a commitment to a message part, such that the commitment hides the message part as long as the randomizer is unknown, i.e. the hiding of redacted message parts is based on the hiding property of the used commitment scheme. It must be mentioned, that one may also use other commitment schemes. However, commitments based on hash functions [20] are usually much more efficient.

In order to be able to build the Merkle-tree to verify the signature for document D, besides the standard signature, the updated signature needs to additionally include the seed of the GGM-tree, i.e. $\sigma_{RS} = (\sigma, r_{v_R})$. The signature verification is straightforward by constructing the GGM-tree by means of the seed r_{v_R}, building the Merkle-tree from the document and verifying the standard signature σ of the hash of the root node. In order to redact block i from D, one deletes the block from D as already mentioned above and updates the signature to be $\sigma'_{RS} = (\sigma, h_{D[i]}, \mathcal{A}_{v_{D[i]}})$ and additionally includes the indices of the redacted parts (which is omitted here). Thereby, the signature contains the randomized hash of the leaf, which does not reveal any information on the removed part and the authentication path $\mathcal{A}_{v_{D[i]}}$ that contains the set of randomizers, which is necessary to reconstruct the remaining part of the GGM-tree. But, the computation of the randomizer $r_{v_{D[i]}}$ for the removed part $D[i]$ is impossible, as long as the used PRG is secure. For notational convenience, we will subsequently denote the set of hash values (commitments) that result from redacted document parts as C and the set of randomizers (in the respective authentication paths) as R. In the example discussed before, our updated signature will be $\sigma'_{RS} = (\sigma, C, R)$, where $C = \{h_{D[i]}\}$ and $R = \mathcal{A}_{v_{D[i]}}$.

5 Generalized Redactable Signatures

Redactable signatures [2] organize the content of an (unstructured) document as leafs of a complete binary tree. For XML documents, however, one can beneficially (cf. section 6) exploit the XML-induced N-ary tree structure to avoid cumbersome assignments to a binary tree *not* resembling the natural structure of the document. For instance, one can construct subtrees from the respective XML-attributes and values.

5.1 Construction

Generalized redactable signatures are obtained by replacing the binary tree construction with a general tree construction. While a Merkle-tree can be defined using general trees rather than binary trees (see equation (2)), the generalization is less obvious for the GGM-tree. This one, in particular the proof of security regarding the outputs, hinges on the binary tree construction.

Our construction that enjoys provable security (since the security is implied by the standard GGM-tree construction) can be described as follows: Suppose we have a tree T, and let v denote an inner node with label r_v and $N > 2$ children. For every child c, we seek to derive a randomizer r_c (to be used in equation (2)) from v's label r_v, such that the children's labels (randomizer) do not provide any information about each other. For the leaves of a (binary) GGM-tree, this has been proven (see [19]). The idea is to simply construct an auxiliary GGM-tree with root v, and (at least) N leafs resembling the children of v. The so-constructed randomizers r for the children of v will enjoy the desired properties, because the leafs of GGM-trees do. Such an auxiliary GGM-tree can be constructed for each inner node of the (general) tree T. The arising additional costs can be considered negligible, since many standard GGM-tree implementations are extremely efficient.

Taking the construction in section 4 as a template, we will devise our scheme for generalized redactable signatures on arbitrary documents D being represented as an N-ary tree. Thereby, we assume that we have already applied a transformation to a document D and for simplicity we will denote a document and its parts as $D = D[1], \ldots, D[n]$, whereas we assume that this is a representation of a N-ary tree. The assignment ϕ for the labels of the nodes of the modified Merkle-tree is adapted to (2), with randomizers r_p created using the generalized GGM-tree construction outlined above. Note, that now also inner nodes hold parts of messages (denoted as x). Note that $C(m, r)$ represents a commitment to message m using randomizer r, e.g. using hash functions [20].

$$\phi(p) = \begin{cases} H(x||\phi(c_0)|| \ldots ||\phi(c_k)) & \text{if } p = \text{root and has } k{+}1 \text{ children,} \\ C((x, \phi(c_0), \ldots, \phi(c_k)), r_p) & \text{if } p \neq \text{root and has } k{+}1 \text{ children,} \\ C(x, r_p) & \text{if } p \text{ is a leaf.} \end{cases} \quad (2)$$

Basically, the computation of the commitments of leave nodes is identical to the Merkle-tree construction and the remaining computations are adapted to N-ary

trees. A generalized redactable signature consists of the signed root value $\phi(p)$ where p is root and the seed r_{v_R} of the root node. More precisely, the signature $\sigma' = (\sigma, C, R)$, where σ is the signature for $\phi(p)$, $C = \emptyset$ and $R = \{r_{v_R}\}$. When redacting a message part, we preceded as in section 4 and if the redacted node is an inner node, the randomizers of all child nodes need to be included.

5.2 Security of Generalized Redactable Signatures

Essentially, the security of generalized redactable signatures has to satisfy two properties, namely *unforgeability* and *hiding*. Hiding refers to the adversary's inability to reconstruct redacted parts of the document. Unforgeability is more involved and requires some definitions [2, 15]: given a document $D = D[1], \ldots, D[n]$ composed of n blocks, a redacted document is $D' = D'[1], \ldots, D'[n]$ with $D'[i] \in \{D[i], \perp\}$ for all $i = 1, 2, \ldots, n$. The more blocks are redacted, the less related the document becomes to its original parent. This is formally modeled through a partial order relation:

Definition 1 *Let $D = D[1], \ldots, D[n]$ and $D' = D'[1], \ldots, D'[n]$ be two (possibly) redacted versions of a document $\mathbf{D}$. We define a partial order $\prec$ on redacted documents such that $D \prec D'$ holds if and only if the following properties are satisfied for all $1 \leq i \leq n$:*

1. *if $D[i] \neq \perp$, then $D'[i] \neq \perp$.*
2. *if $D[i] = \perp$, then either $D'[i] = \mathbf{D}[i]$ or $D'[i] = \perp$.*

Roughly speaking, unforgeability prevents the adversary from finding a signature for a document D such that $D' \prec D$ for a given and signed document D'. In contrast to security definitions of standard signature schemes [21], we are not focusing on existential unforgeability against chosen message attacks (eUF-CMA). Recall that in the game based proof for eUF-CMA security, an adversary can issue signature queries for adaptively chosen messages of his choice, and finally needs to output a valid signature for a message that has not been issued to the sign oracle during the game. But we will base the security on a game capturing the aforementioned unforgeability and hiding properties at the same time and furthermore relate the signature forgery to a redacted document of the adversary's choice. Nevertheless, is should be noted that we require the standard signature scheme used to sign the root of the modified Merkle-tree to provide eUF-CMA security. Thereby, our game is based on the security model of [15]. Basically, an adversary who manages to win this game with non-negligible probability, will be able to breach the security of some of the used cryptographic primitives with non-negligible probability. Our definition of secure generalized redactable signature schemes follows the same ideas and is as well based on a game in the aforementioned sense.

Definition 2 *A generalized redactable signature scheme is said to be secure, if no probabilistic polynomial-time adversary has a non-negligible advantage against the challenger in game 5.2.*

Game 5.2 The following oracles are available to the adversary:

1. commit: A commit query is defined as issuing a document D to the oracle, which chooses a random seed r_{v_R}, derives randomizers for all nodes of the N-ary tree (assume that the tree has n nodes), computes the commitments h_{v_i}, $1 \leq i \leq n$, for all nodes and returns the tuple $(r_1, \ldots, r_n, h_{v_1}, \ldots, h_{v_n})$.
2. sign: A sign query is defined as issuing the hash value of the root node h_{v_R} to the oracle, which uses the private-key SK of a eUF-CMA secure digital signature scheme, computes a signature $\sigma = S_{SK}(h_{v_R})$ and returns σ.
3. redact: A redact query is defined as issuing a generalized redactable signature σ_{GRS}, a document D and a set of indices (which defines the document parts to be redacted) to the oracle, which computes the redaction and returns the redacted document D' along with an adapted generalized redactable signature σ'_{GRS}.

The adversary plays the following game for breaking the signature scheme:
Setup: The challenger takes a security parameter, chooses a secure digital signature scheme, generates a key-pair (SK, PK) for this scheme, chooses a cryptographic hash function H as well as a commitment scheme C for the tree computation and chooses a PRG G to generate the randomizers, gives (PK, H, C, G) and the signature scheme description to the adversary and keeps SK secret.
Phase 1: The adversary issues queries $q_1, \ldots, q_m$, where $q_i \in \{\mathsf{commit}, \mathsf{sign}, \mathsf{redact}\}$ and these queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \ldots, q_{i-1}$.
Challenge: The adversary may want to attack the hiding property (figure out redacted parts of the document) by providing two documents D_0 and D_1 of equal length along with a list of parts to be redacted to the challenger, which differ in exactly one document part, i.e. for exactly one i it holds that $D_0[i] \neq D_1[i]$. The challenger chooses a random bit b, produces a generalized redactable signature for D_b and redacts the document (the i'th document part must be redacted).
Phase 2: The adversary issues additional queries $q_{m+1}, \ldots, q_n$ where query $q_i \in \{\mathsf{commit}, \mathsf{sign}, \mathsf{redact}\}$ and these queries may be asked adaptively. Thereby, the adversary is not allowed to ask any sign or redact queries for any $D_b \prec D$.
Guess: The adversary outputs a guess for either attacking the hiding or the unforgeability property.
Hiding: The adversary outputs a bit b' and wins the game if $b = b'$ holds. The adversary's advantage is defined as $|\mathsf{Pr}[b = b'] - \frac{1}{2}|$.
Unforgeability: The adversary outputs a generalized redactable signature $\sigma' = (\sigma, C, R)$ for a document D, whereas no sign or redact query has been issued to any document D' such that $D \prec D'$ or $D_b \prec D'$. The adversary's advantage is defined as the probability that the verification of the signature succeeds when the two aforementioned conditions hold.

Theorem 1 (proof appears in appendix A). *If the signature schemes provides eUF-CMA security, H is a collision-resistant cryptographic hash function, C a secure commitment scheme and G a secure PRG, then the generalized redactable signature scheme is a secure one.*

6 Extensions to Generalized Redactable Signatures

In this section we present extensions to realize *controlled replacement* and *fine-grained redaction*. Redactable signatures in the sense above allow for "deleting" parts of the document by replacing them with the empty symbol $\perp$. However, it might be interesting for the signer to specify a set of permitted replacements for certain parts of the document. An authentic (signed) questionnaire would be an example of such a scenario. Sanitizable signatures permit replacements of *designated* parts by *designated* redactors, who are free to substitute any string they like. Recently, [11] introduced *controlled* replacements. These limit the redactors substitution options to a set S that can be specified by the verifier. The idea is to replace the empty symbol $\perp$ by an element $x \in S$ along with a compact representation of the set S of permissible substitutions. Bloom-Filters [22] or accumulators [23] are appropriate for that matter. Testing for a given substitution is easy by querying the Bloom-Filter or the accumulator (notice that a small probability for getting a false-positive answer is inevitable in case of Bloom-filters). Both approaches are deterministic and as such allow for brute-force identification of the set S, if it is sufficiently small.

We sketch two remedies for this drawback: *randomized accumulators* and *generalization hierarchies*. The secure RSA-based accumulator, specified in [16], employs hash-functions. Replacing those hash-values by message-authentication codes with secret random keys, creates a *randomized accumulator* from a standard (deterministic) accumulator. Simply replace the hash-function $H(s)$ by a randomized commitment for s based on hash functions, i.e. by $H(s, r_s)$, where r_s is a randomizer that is *specific* for s. Providing this randomizer in addition permits querying whether s is contained in the accumulator, but prevents brute-force searching for other elements s', because their specific randomizers (keys) $r_{s'}$ remain unknown. A randomizer r_s for s can be derived using GGM-trees.

Generalization hierarchies are particularly suited for k-anonymity and generalization sets. As an example, consider an attribute ZIP= 22047 and its associated generalization set $S = \{22047, 2204*, 220**, 22***\}$. The ZIP-code shall only be replacable by elements of S. Simply change this block into the root of a subtree with leafs being the members of S. For the example, the child nodes would have labels 22, 0, 4 and 7. Redactions work in the obvious way.

7 Conclusion

In this paper we have presented a generalization of redactable signature schemes and several extensions. In contrast to standard redactable signatures, our scheme takes advantage of the inherent structure of structured documents like XML-documents. This is on the one hand a more natural way to access this problem and on the other hand beneficial for extensions like generalization hierarchies. Furthermore, the generalization induces only little additional computational overhead, which we attempt to confirm upon implementations. We believe

that this concept is a valuable tool in the context of electronic healthcare, since it combines redaction capability with verification of the authenticity of remaining data while improving the efficiency of medical treatment processes. Future research includes further development of the proposed extensions, along with respective implementations (including an analysis of how much the XML structure affects the scheme's efficiency). A performance study to evaluate the quantitative benefit in terms of storage space and computational effort is part of ongoing work.

References

1. Steinfeld, R., Bull, L., Zheng, Y.: Content Extraction Signatures. In: ICISC 2001. Volume 2288 of LNCS., Springer (2001) 285–304
2. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic Signature Schemes. In: CT-RSA '02. Volume 2271 of LNCS., Springer (2002) 244–262
3. Merkle, R.: A Certified Digital Signature. In: CRYPTO '89. Volume 435 of LNCS., Springer (1989) 218–238
4. Golle, P.: Revisiting the Uniqueness of Simple Demographics in the US Population. In: WPES 2006, ACM (2006) 77–80
5. Ciriani, V., di Vimercati, S.D.C., Foresti, S., Samarati, P.: Theory of Privacy and Anonymity. In: Algorithms and Theory of Computation Handbook. CRC Press (2009)
6. Samarati, P.: Protecting Respondents' Identities in Microdata Release. IEEE Trans. Knowl. Data Eng. **13**(6) (2001) 1010–1027
7. Sweeney, L.: k-Anonymity: a Model for Protecting Privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **10**(5) (2002) 557–570
8. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schroeder, D., Volk, F.: Security of Sanitizable Signatures Revisited. In: PKC 2009. LNCS, Springer (2009)
9. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: 10th European Symp. on Research in Computer Security - ESORICS 2005. Volume 3679 of LNCS., Springer (2005) 159–177
10. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor Sanitizable Signatures and Their Application to Content Protection. In: ACNS '08. Volume 5037 of LNCS., Springer (2008) 258–276
11. Klonowski, M., Lauks, A.: Extended Sanitizable Signatures. In: ICISC 2006. Volume 4296 of LNCS., Springer (2006) 343–355
12. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: EUROCRYPT 2003. Volume 2656 of LNCS., Springer (2003) 416–432
13. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally Signed Document Sanitizing Scheme Based on Bilinear Maps. In: Proc. of the 2006 ACM Symp. on Information, Computer and Communications Security, ASIACCS 2006, ACM (2006) 343–354
14. Izu, T., Kunihiro, N., Ohta, K., Sano, M., Takenaka, M.: Yet Another Sanitizable Signature from Bilinear Maps. In: ARES 2009, IEEE Computer Society (2009) 941–946
15. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient Signature Schemes Supporting Redaction, Pseudonymization, and Data Deidentification. In: Proc. of the 2008 ACM Symp. on Information, Computer and Communications Security, ASIACCS 2008, ACM (2008) 353–362

16. Chang, E.C., Lim, C., Xu, J.: Short Redactable Signatures Using Random Trees. In: CT-RSA '09. Volume 5473 of LNCS., Springer (2009)
17. Nojima, R., Tamura, J., Kadobayashi, Y., Kikuchi, H.: A Storage Efficient Redactable Signature in the Standard Model. In: ISC 2009. Volume 5735 of LNCS., Springer (2009) 326–337
18. Bauer, D., Blough, D., Mohan, A.: Redactable Signatures on Data with Dependencies and their Application to Personal Health Records. In: Proc. of the 8th ACM Workshop on Privacy in the Electronic Society, WPES '09, New York, NY, USA, ACM (2009) 91–100
19. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. J. ACM **33**(4) (1986) 792–807
20. Halevi, S., Micali, S.: Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In: CRYPTO '96. Volume 1109 of LNCS., Springer (1996) 201–215
21. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. on Computing **17**(2) (1988) 281–308
22. Bloom, B.: Space/Time Trade-offs in Hash Coding with Allowable Errors. Commun. ACM **13**(7) (1970) 422–426
23. Benaloh, J., de Mare, M.: One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract). In: EUROCRYPT '93. Volume 765 of LNCS., Springer (1993) 274–285

A Proof of Theorem 1

For the proof of Theorem 1 we use collision-resistant hash functions H, a secure commitment scheme C, a secure PRF G and a $\mathsf{eUF\text{-}CMA}$ secure digital signature scheme. We omit the formal definitions here, but note that for collision-resistant hash functions the probability of finding two messages $m \neq m'$ with $H(m) = H(m')$ is negligible and, for a secure commitment scheme only with negligible probability, one will be able to figure out m from $C(m, r)$ and to open $C(m, r)$ to some $m' \neq m$.

Basically, the proof follows the reduction to contradiction paradigm, i.e. we construct an adversary $\mathcal{B}$ that uses the adversary in game 5.2 (called $\mathcal{A}$) whereas $\mathcal{B}$ plays the role of the challenger of $\mathcal{A}$ (using $\mathcal{A}$'s output as his own) and can break one of the used cryptographic primitives with non-negligible advantage.

Proof (sketch): Subsequently, we sketch how $\mathcal{B}$ acts when using adversary $\mathcal{A}$:
Setup: $\mathcal{B}$'s challenger runs the setup, gives (PK, H, C, G) as well as the description of the signature scheme to $\mathcal{B}$. $\mathcal{B}$ gives all the information to $\mathcal{A}$ (note that $\mathcal{A}$ and $\mathcal{B}$ do not know SK).
Phase 1: $\mathcal{A}$ is run and issues queries $q_1, \ldots, q_m$ as in game 5.2, but issues his queries $q_i \in \{\mathsf{commit}, \mathsf{sign}, \mathsf{redact}\}$ to $\mathcal{B}$. Except for queries of type sign, $\mathcal{B}$ acts as defined in section 5.2. However, since $\mathcal{B}$ does not know SK he forwards these types of queries to his challenger.
Challenge: $\mathcal{A}$ may want to attack the hiding property and outputs two equal length documents D_0 and D_1, which differ in exactly one document part, i.e. for exactly one i it holds that $D_0[i] \neq D_1[i]$, along with a list of parts to be

redacted. Now, $\mathcal{B}$ will use the advantage of $\mathcal{A}$'s guess (in the guess phase) to break the hiding property of the commitment scheme C. He proceeds as follows: He takes D_0 and D_1 as the two messages for breaking the hiding property of C and gives them along with i to his challenger. The challenger chooses at random a bit b and computes a commitment $h_{D_b[i]}$ for $D_b[i]$ (note that if $D_b[i]$ is an inner node, the challenger needs to compute the hash of the root of this subtree by means of a commit query). $\mathcal{B}$ uses $h_{D_b[i]}$ and computes the Merkle-tree using the remaining document (note that the remaining parts of D_0 and D_1 are identical). Now $\mathcal{B}$ has embedded his challenge (to attack C) into $\mathcal{A}$'s challenge and issues a sign query on the root hash to his challenger. Hence, he obtains a generalized redactable signature, redacts at least the i'th subdocument and gives the result back to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ issues queries $q_{m+1}, \ldots, q_n$ as in phase 1 to $\mathcal{B}$ (with the same restrictions as defined in section 5.2).
Guess: $\mathcal{A}$ outputs a guess for either attacking the hiding or the unforgeability property and $\mathcal{B}$ uses this guess for breaking one of the primitives (signature scheme, hash function, commitment scheme, PRG).
Hiding: Adversary $\mathcal{A}$ outputs a guess b which is used by $\mathcal{B}$. Clearly, if $\mathcal{A}$ wins, then $\mathcal{B}$ is able to break the hiding property of the commitment scheme C. Consequently, if $\mathcal{A}$'s advantage in winning is non-negligible, $\mathcal{B}$ will break the commitment scheme with non-negligible probability, which contradicts our assumption.
Unforgeability: Adversary $\mathcal{A}$ outputs a generalized redactable signature (as defined in game 5.2). Now we need to investigate all possible cases: Let us assume that this signature is not equal to any of the signatures returned by $\mathcal{A}$ during his game, i.e. $\mathcal{A}$ has broken the unforgeability property. But, now $\mathcal{B}$ could construct the Merkle-tree for the corresponding document and the root hash will be $h_{v_R} = H(x||\phi(c_0)||\ldots||\phi(c_k))$ which is a hash of a possible label x of the root node and a concatenation of the values of the $k+1$ children (as commitments to their subtrees). But the signature constitutes a signature of exactly these values, which can be used by $\mathcal{B}$ to break the existential unforgeability property of the respective signature scheme. The remaining case is a signature that is identical to a signature which was output by $\mathcal{A}$ during one of the phases. Firstly, either the two Merkle-trees constructed for the respective documents differ in their roots, then $\mathcal{B}$ has found a signature forgery for the digital signature scheme. Otherwise, the two trees differ anywhere below the root node and $\mathcal{B}$ has found a collision for the hash function H. Secondly, $\mathcal{B}$ investigates the construction of auxiliary GGM-tree (note that the security of this construction is implied by that of the standard GGM-tree). If there exists an index j where both documents differ, then he has found two document parts which can be used to break the hiding property of C. Finally, we need to look at the restriction, whereas we denote the two documents as D and D'. We have required that $D \prec D'$ needs to hold. But if $\mathcal{A}$ outputs as a guess a signature for a document D that contains some position that has been redacted in D' (in one of the two phases) and is present in D, then $\mathcal{B}$ can use this to break the hiding property of C or the PRG G underlying the auxiliary GGM-tree construction. The proof is complete.