Redactable Graph Hashing, Revisited

Andreas Erwig   Marc Fischlin   Martin Hald   Dominik Helm   Robert Kiel
Florian Kübler   Michael Kümmerlin   Jakob Laenge   Felix Rohrbach

Technische Universität Darmstadt, Germany

Abstract. We revisit the previous work of Arshad et al. (CODASPY 2014) about the security of redactable graph hashing schemes. Such schemes, introduced in a series of works by Devanbu et al. (DBSec 2000, CCS 2001, Algorithmica 2004), allow to hash graphs and to release sub graphs which can be verified against the original hash value. Arshad et al. introduce security notions for collision resistance and privacy of graphs, where the latter should capture the infeasibility to reconstruct the full graph from the hash value of a redacted one.

We discuss here that the original security notions of Arshad et al. are too weak. Our argument is by virtue of intuitively insecure examples which are deemed secure according to their notion. We therefore present stronger security definitions. We also point out the differences in the privacy notions with respect to redactable and sanitizable schemes: In the former case anyone can produce verifiable data from the graph, whereas in the latter case only a designated party can. Sanitizable schemes allow for stronger privacy guarantees. We finally discuss instantiation possibilities for the various security notions.

1 Introduction

Cryptographic primitives are often used to protect static data. But with the growth of outsourcing computations and data maintenance, the need to have primitives supporting operations on the secured data has also increased. For instance, the breakthrough construction of fully homomorphic encryption [15] allows in principle to run now computations on encrypted data, ensuring the privacy of the data towards the evaluating party. Even earlier, for authenticity and integrity the ideas of redactable or sanitizable signature schemes [24, 18, 2] have introduced the possibility to sign data in such a way that external parties can prove authenticity of partial data. This may require to protect the privacy of the redacted data, e.g., when handing out partial medical data [3, 25, 5, 4].

1.1 Redactable Graph Hashing

In this work we look at the notion of redactable graph hashing. The idea is that one can create a hash value of a graph such that one can later verify the hash against any (redacted) sub graph, yet possibly requiring some additional information for the verification. The idea has been introduced by Martel et al. [20] for (directed) acyclic graphs, with a focus on designing solutions. Recently, Arshad et al. [1] extended the approach to cyclic graphs and augmented the security considerations by more formal definitions and claims.

The approach of redactable graph hashing is highly convincing for designing functional cryptographic schemes. First, graphs are very general concepts such that devising constructions for graphs immediately gives solutions for a variety of other data structures. Second, redactable hash functions instantaneously yield redactable signature schemes. For this, one merely applies the common hash-then-sign paradigm, where the signer signs the graph hash value with a regular signature scheme. Redaction on the graph, which supposedly leaves the hash value untouched, then does not require to change the signature part for updating the cryptographic data.

1.2 Defining Security

Arshad et al. [1] present security definitions for collision-resistance and privacy of redactable graph hashing, as well as constructions and performance results. The former security property should guarantee that one cannot efficiently find distinct graphs with identical hash values, and the latter one should prevent leakage of information about the redacted parts. While Arshad et al. clearly deserve credit for putting forward the formal requirements of such schemes, our starting point is to note that their formal security notions do not seem to appropriately capture the desired properties. For instance, we point out that their definition of collision resistance only captures adversaries which faithfully create hash values and redactions according to the scheme. In general, however, adversaries may choose such data maliciously, and we indeed present a scheme which is intuitively insecure, but provides collision resistance according to their definition.

We therefore present new definitions for the security properties, following similar approaches for redactable and sanitizable signatures [7, 8]. Our notion of collision resistance of graph hashing demands that the adversary cannot output a hash value and different graphs G, G' such that they both verify against the hash value. Of course, neither of the graphs can be a sub graph of the other one, since otherwise an adversary could simply redact G to G' to get different graphs for the same hash value. But the exact formalization is even a bit more tricky, since the adversary may derive both G and G' form a common super graph with the same hash value.

Another shortcoming in [1] refers to the privacy notion. If one adopts the common idea of privacy from the domain of redactable signatures, then privacy should guarantee that one cannot deduce any information about the original graph from the redacted version. We demonstrate that the privacy definition in [1] does not capture this property. That is, we present a scheme where redaction clearly leaks information about the original graph, but is deemed secure according to their notion. We therefore give a new definition of privacy in the spirit of redactable and sanitizable signatures.

1.3 Redactable Graph Hashes, or Sanitizable Graph Commitments?

An important conceptual observation we make here is that graph hashing may come in two flavors. One flavor follows the idea of hashing more closely, and assumes that anyone can re-hash the graph in question and check the graph against a given hash value. This usually assumes that the randomness in the hashing step, if any at all, is made public. Redaction can then be performed by anyone. The other option is to view the hash of a graph rather as a commitment, involving some secret randomness. This means that only a designated party, usually called the sanitizer, can use the secret randomness to provide a verifiable proof for redacted graphs.

Both approaches, redactable graph hashing and sanitizable graph commitments, are valid strategies in order to give out partial and authenticated information about the full graph. In both cases the owner of the graph may publish (a signed version of) the hash value or commitment, and subsequently give out sub graphs, whose correctness with respect to the initial value can be verified with the help of additional data. In the sanitization case, however, this step can be only done by a designated party holding some auxiliary secret data. Indeed, Martel et al. [20] implicitly consider this option when they speak of a publisher for the authenticated partial data, and made this even more explicit in a previous works [13, 12, 14]. We note that Arshad et al. [1] purely consider graph hashing schemes.

The advantage of the commitment approach is that it facilitates the hiding of the redacted parts of the graph. In the graph hashing setting any reasonable notion of privacy, arguing that the adversary cannot deduce information about the original graph from seeing a redacted version, inevitably requires some non-trivial form of uncertainty (i.e., entropy) in the graph. The reason is that the adversary could otherwise determine the full graph by matching the given hash value against (publicly computable) hash values of potential super graphs.

The terminology in our paper will be general enough to capture both cases simultaneously. Only for privacy we need to make a slight distinction. For sake of simplicity we will subsume both notions under the term redactable graph hashing.

1.4 Constructions

Finally, we show that our security properties can be met. Our construction follows the approach of Arshad et al. [1] by decomposing the graph into nodes and edges, and hashing all these components individually with a cryptographic hash function. Yet, while the approach in [1] requires a special treatment of edges violating the common tree structure, we do not make a fine-grained distinction.

Interestingly, our basic construction works for all of the aforementioned security properties by switching to a different component for the cryptographic hash function. If one is only interested in collision resistance, but not privacy, for the redactable hashing scheme, then a common collision-resistant cryptographic hash function for the individual hashes suffices. If one is interested in privacy for the hashing scheme, then we show that a strong form of perfectly one-way hash function [9, 10] works. We also argue that this function can be built easily from a random oracle. Finally, if one is interested in privacy for a sanitizable graph commitment scheme, then one can use a non-interactive commitment scheme for processing the nodes and edges. The commitment in turn can be derived from a collision-resistant hash function. In all cases we prove the constructions to be secure according to our notions.

In all cases, however, the size of a hash becomes quite large. This also holds for the solution in [1]. We therefore also discuss how one can shrink the size by using Merkle hash trees [21] with a collision-resistant hash function. This version preserves our corresponding security properties.

1.5 Comparison to Related Concepts

As mentioned before, our work is inspired by the work of Arshad et al. [1] about security notions for redactable hashing of general graphs. This work itself is based on preliminary work in this area of Martel et al. [20] and Devanbu et al. [13, 12, 14]. The idea of being able to redact (or to sanitize) structured data appears in many different contexts, e.g., [19, 6, 16, 22, 23].

The idea of redactable graph hashing is clearly related to the notion of redactable signature schemes. Such signature schemes, however, often operate on strings and message blocks instead of graphs. Still they, too, often apply ideas like Merkle hash trees for constructions. Yet, they usually do not consider collision resistance (of graph hashing) as an abstract security goal itself and thus do not define this property. Also, when processing strings, there is no need to distinguish explicitly between privacy of the structure and privacy of the content.

The idea of sanitizable graph commitments is related to the notion of sanitizable signatures. The same discussion about the difference between redactable graph hashes and redactable signature schemes applies here: Collision resistance and privacy of the more complex structure are usually not considered for sanitizable signatures. For sanitizable graph commitments we do not consider additional security properties sometimes discussed for sanitizable sigantures. This refers to properties such as unlinkability, an even stronger form of privacy guaranteeing that one cannot link redactions of the same graph, or accountability, saying that one can provide a proof who sanitized the graph.

2 Preliminaries

Graphs. A directed labeled graph $G = (V, E, \textsc{content})$ is a set of nodes (or vertices) V and a set of edges $E \subseteq V \times V$, where each edge has a source and a destination node. The content (or labeling) function $\textsc{content} : V \cup E \to \mathcal{C}$ maps the nodes and edges to some string in the content space $\mathcal{C} \subseteq \{0,1\}^*$. In the following we sometimes write $e(u, v)$ to denote the edge e with source node u and destination node v. We usually denote the number of nodes by $|V| = n$ and the number of edges by $|E| = m$. A sub graph $G_{\mathrm{sub}} = (V_{\mathrm{sub}}, E_{\mathrm{sub}}, \textsc{content}_{\mathrm{sub}})$ of G, written $G_{\mathrm{sub}} \subseteq G$, is itself a graph and satisfies $V_{\mathrm{sub}} \subseteq V$ and $E_{\mathrm{sub}} \subseteq E \cap V_{\mathrm{sub}} \times V_{\mathrm{sub}}$, as well as $\textsc{content}_{\mathrm{sub}}(x) = \textsc{content}(x)$ for all $x \in V_{\mathrm{sub}} \cup E_{\mathrm{sub}}$.

For hashing the graph it is convenient to associate an absolute order on the nodes and edges. That is, we assume that there exists an implicit injective mapping $\textsc{order} : V \cup E \to \{1, 2, \ldots, m + n\}$. For example, for some order on the nodes $v_1, \ldots, v_n$ and the edges $e_1, \ldots, e_m$, e.g., according to their position in the digital representation, the function could be defined by $\textsc{order}(v_i) = i$ and $\textsc{order}(e_i) = n + i$. This ordering also allows us to identify each node and edge with a number between 1 and $m + n$. In addition, for privacy reasons we also use a random order in the sense that we introduce a random permutation $\pi : \{1, 2, \ldots, m + n\} \to \{1, 2, \ldots, m + n\}$. Composing this with the ordering $\textsc{order}$ this gives a bijection $\textsc{order}_\pi = \pi \circ \textsc{order}$ from $V \cup E$ to $\{1, 2, \ldots, m + n\}$.

Redactable Graph Hashing. Recall that we treat both redactable graph hashes as well as sanitizable graph commitments integratively, and only speak of redactable graph hashes. The difference between the two cases shows in the approach below when already the hashing algorithm outputs some secret information vo, called verification object in [20, 1], necessary to create verification data vo_{sub} for a sub graph G_{sub}.

The key generation and hashing algorithms HKGen and Hash follow the common approach for defining hash functions. In addition, we introduce a redaction algorithm $\mathsf{HRedact}$. Since we consider randomized outputs for both hash values and commitments, we cannot necessarily recompute the hash value of some input and compare it to a given value. We therefore more abstractly introduce a verification algorithm HVf which checks the validity of a hash value. To define a reasonable notion of collision-resistance later, we need to distinguish the cases that the verification algorithm HVf checks for a full hash value or for a redacted value, and hence pass the operation mode $\mathsf{vfmode} \in \{\mathsf{hashed}, \mathsf{redacted}\}$ as additional input to HVf.

Definition 2.1 (Redactable Graph Hashing) *A redactable graph hashing scheme $\mathcal{H} = (\mathsf{HKGen}, \mathsf{Hash}, \mathsf{HRedact}, \mathsf{HVf})$ consists of four probabilistic polynomial-time algorithms:*

Key Generation: *The key generation algorithm, on input the security parameter 1^n, outputs a public hash key, $hk \leftarrow_\$ \mathsf{HKGen}(1^n)$.*

Hashing: *On input the hash key hk and a graph G, the (probabilistic) hashing algorithm returns a hash value, together with a (potentially empty) verification object, $(gh, vo) \leftarrow_\$ \mathsf{Hash}(hk, G)$.*

Redaction: *On input the hash key hk, a hash value gh, a graph G, a sub graph $G_{sub} \subseteq G$, and possibly empty data vo, the (probabilistic) redaction algorithm returns a proof, $vo_{sub} \leftarrow_\$ \mathsf{HRedact}(hk, gh, G, G_{sub}, vo)$.*

Verification: *On input the hash key hk, a hash value gh, a graph G, a (potentially empty) proof vo, and a mode identifier $\mathsf{vfmode} \in \{\mathsf{hashed}, \mathsf{redacted}\}$, the verification algorithm returns a decision bit, $d \leftarrow_\$ \mathsf{HVf}(hk, gh, G, vo, \mathsf{vfmode})$.*

We assume the usual correctness property that genuine hash values of graphs are accepted, i.e., for any security parameter n, any $hk \leftarrow_\$ \mathsf{HKGen}(1^n)$, any graph G, and any hash value $(gh, vo) \leftarrow_\$ \mathsf{Hash}(hk, G)$, we have $\mathsf{HVf}(hk, gh, G, vo, \mathsf{hashed}) = 1$ with probability 1. Furthermore, the same holds recursively for properly redacted values, i.e., for any n, any $hk \leftarrow_\$ \mathsf{HKGen}(1^n)$, any triple (gh, G, vo) with $\mathsf{HVf}(hk, gh, G, vo, \mathsf{vfmode}) =$

$\mathbf{Exp}^{\mathrm{CR}}_{\mathcal{H},\mathcal{A}}(1^n)$

```
1 :  $hk \leftarrow_\$ \mathsf{HKGen}(1^n)$
2 :  $(gh, G, G', vo, vo') \leftarrow_\$ \mathcal{A}(hk)$
3 :  return $1$ if
4 :     $G' \not\subseteq G$ and $\mathsf{HVf}(hk, gh, G, vo, \mathsf{hashed}) = 1$
5 :     and $\mathsf{HVf}(hk, gh, G', vo', \mathsf{vfmode}') = 1$ for some $\mathsf{vfmode}' \in \{\mathsf{hashed}, \mathsf{redacted}\}$
```

Figure 1: Collision-Resistance Experiment for Graph Hashing

1 *for some* $\mathsf{vfmode} \in \{\mathsf{hashed}, \mathsf{redacted}\}$*, for any* $G_{sub} \subseteq G$ *and any* $vo_{sub} \leftarrow_\$ \mathsf{HRedact}(hk, hval, G, G_{sub}, vo)$*, we also have* $\mathsf{HVf}(hk, gh, G_{sub}, vo_{sub}, \mathsf{redacted}) = 1$ *with probability* 1*.*

3 Security Properties

In this section we define our notions of collision resistance and of privacy. The latter comes in two flavors, depending on whether one considers graph hashing or graph commitments. We also scrutinize the shortcomings of the definitions in [1] at the end of this section.

3.1 Collision Resistance

The underlying idea behind defining collision resistance is that the adversary should neither be able to find different graphs which verify under the same hash value, nor to make the verifier accept a graph which is not a sub graph of the graph belonging to the hash value. The latter already includes the case of different graphs, such that there is no need to distinguish the two events below.

The redaction property introduces some additional complications with the above approach. Assume that the adversary creates the hash value gh for some graph G, and then redacts this graph with the hash value twice, for two distinct sub graphs $G_{\mathrm{sub}}, G'_{\mathrm{sub}}$. By construction both graphs would result in the same hash value gh, only the verification objects would differ, but they are a means to an end, similar to the randomizer for computing the hash value.

One option to prevent "trivial" redaction attacks would be to declare the adversary to lose if there exists a common super graph to the colliding graphs. But there may always exist such a graph for the hash function with shrinking output, such that we rather ask the adversary to specify the super graph. That is, we let the adversary win if one of the graphs verifies as a full hash, and the other graph may either verify as a full hash or a redacted one. In any case, the second graph must not be a sub graph of the former one.

To given an argument for the appropriateness of our notion of collision resistance consider once more the setting where a party outputs the (signed) hash as a commitment to the full graph G. Then the party should not be able to later present a graph for the hash value which is not a sub graph of G. This is indeed captured by our notion of collision resistance.

We give our definition in terms of asymptotic security; a concrete statement is easy to deduce.

Definition 3.1 (Collision-Resistance) *A redactable graph hashing scheme* $\mathcal{H}$ *is collision-resistant if for any probabilistic polynomial-time algorithm* $\mathcal{A}$ *the probability*

$$\mathrm{Prob}\left[\boldsymbol{Exp}^{CR}_{\mathcal{H},\mathcal{A}}(1^n)\right] \approx 0$$

for the experiment $\boldsymbol{Exp}^{CR}_{\mathcal{H},\mathcal{A}}(1^n)$ *in Figure 1 is negligible.*

3.2 Privacy

Privacy can be divided according to the two views on graph redaction schemes, one time if viewed as a hash value, and the other time if seen as a commitment. The difference between the privacy is roughly as follows: Suppose that you see a redacted graph and its hash value *gh*. In the hashing setting, if you have high confidence that the graph originates from a super graph then you can check the super graph against the public hash value *gh* to achieve certainty. In the commitment case, however, you would not be able to decide between two possible super graphs, since you cannot recompute the commitment without knowledge of the secret randomness. We hence need to distinguish between privacy notions for the two settings.

Content-Privacy for Hash Values. For the hash scenario we follow the approach of Canetti et al. [9, 10] of so-called perfectly one-way hash functions. Arshad et al. [1] have also based their definitions on this idea. Such hash functions assume that the input value x carries enough entropy such that searching for the right pre-image to the (randomized) hash value $(r, h(y; r))$ is infeasible. The hiding property of such hash functions demands that t hash values of the same input y (with fresh randomness $r_1, \ldots, r_t$) are indistinguishable from t hash values of completely independent inputs (y_i, r_i). Canetti et al. [10] show that for $t = 2$ this already implies "semantic" privacy in the sense that, from a hash value, one cannot non-trivially compute anything about the pre-image.

We call our notion for redactable graph hashing here *content privacy*, because it relies on the individual entropy in the $\textsc{content}$ value of nodes and edges and does not aim to hide the structure of the graph. Intuitively, the following notion of non-trivial entropy says that one cannot predict efficiently the content of a redacted element, even if one sees all the content of the rest of the graph.[1] Formally, we say that a graph *G has non-trivial entropy with respect to a sub graph G_{sub}* if for each element $x \in (V \cup E) \setminus (V_{\text{sub}} \cup E_{\text{sub}})$ the conditional min-entropy

$$H_\infty(\textsc{content}(x) | \textsc{content}((V \cup E) \setminus \{x\}) = \omega(\log n)$$

is super-logarithmic in the security parameter n. Here the probability is over the choice of (the randomness for) $\textsc{content}$, and $\textsc{content}((V \cup E) \setminus \{x\})$ is the sequence $(y, \textsc{content}(y))$ for all elements y in $(V \cup E) \setminus \{x\}$.

To define privacy we assume that the adversary picks a (sub) graph G_{sub} and an efficiently samplable distribution $\mathcal{G}$ which generates super graphs of G_{sub} such that the generated graph $G \leftarrow \mathcal{G}(1^n)$ always satisfies $G_{\text{sub}} \subseteq G$, G has a non-trivial entropy with respect to G_{sub}, and all graphs G have the same super sets V and E. We call such distributions $\mathcal{G}$ *well-formed* with respect to G_{sub}. Then, the adversary either gets to see t hashes of the same random graph G, or we generate t independent graphs $G_i \leftarrow \mathcal{G}(1^n)$ and return hashes of these graphs. In the definition below we assume that t is a fixed polynomial in the security parameter. The adversary's goal is to decide which kind of hashes the challenger has created.

Definition 3.2 (Content-Privacy) *A redactable graph hashing scheme $\mathcal{H}$ is t-content private if for any probabilistic polynomial-time algorithm $\mathcal{A}$ the probability*

$$\text{Prob}\left[\boldsymbol{Exp}_{\mathcal{H},\mathcal{A},t}^{ContPriv}(1^n)\right] \approx \frac{1}{2}$$

for the experiment $\boldsymbol{Exp}_{\mathcal{H},\mathcal{A},t}^{ContPriv}(1^n)$ in Figure 2 is negligibly close to $\frac{1}{2}$.

Note that, by construction, the well formedness of the distribution $\mathcal{G}$ ensures that all potential super graphs have the same sets V and E, such that our privacy experiment does not aim to hide the structure of the original graph.

[1]One could merely ask for non-trivial entropy in all redacted elements together, but leaving the choice about the sub graph to the adversary, and in particular over all sub graphs with one element missing, implies the above notion of individual entropy.

```
$\mathbf{Exp}^{\mathrm{ContPriv}}_{\mathcal{H},\mathcal{A},t}(1^n)$
1 : $hk \leftarrow\!\!\$\ \mathsf{HKGen}(1^n)$
2 : $(\mathtt{st}, G_{\mathrm{sub}}, \mathcal{G}) \leftarrow\!\!\$\ \mathcal{A}(hk)$   // distribution $\mathcal{G}$ must be well-formed
3 : $b \leftarrow \{0,1\}$
4 : if $b = 0$ then
5 :   $G \leftarrow \mathcal{G}(1^n)$
6 :   for $i = 1..t$ do $(gh_i, vo_i) \leftarrow \mathsf{Hash}(hk, G)$ endfor
7 : else
8 :   for $i = 1..t$ do
9 :     $G_i \leftarrow \mathcal{G}(1^n)$
10 :     $(gh_i, vo_i) \leftarrow \mathsf{Hash}(hk, G_i)$
11 :   endfor
12 : fi
13 : $a \leftarrow \mathcal{A}(\mathtt{st}, gh_1, vo_i, \ldots, gh_t, vo_t)$
14 : return 1 if $a = b$
```

Figure 2: Content-Privacy Experiment for Graph Hashing. A distribution $\mathcal{G}$ is well-formed with respect to G_{sub} if it generates only super graphs of G_{sub} which have the same super sets V and E, and which have non-trivial entropy with respect to G_{sub}.

Also observe that we assume that edges are labeled, too, and that the content of a redacted edge has non-trivial entropy (even if one is given the contents of the start and end node). This is necessary for a general definition, since the redacted part could only remove some edges. In this case if there was no uncertainty in the edge, e.g., if its content is defined by the content of the two nodes, then the adversary could again check whether the edge has been there or not.

Graph-Privacy for Commitments. Next we look at the privacy of graph commitments. Here we drop the requirement on the entropy of the graph, and instead assume that the adversary chooses $G_{\mathrm{sub}}, G_0, G_1$ at will, with the only stipulation that $G_{\mathrm{sub}} \subseteq G_0, G_1$ and that G_0, G_1 have the same number of nodes and edges. The adversary then gets to see the commitment to either of the graphs, but not the verification objects vo. The adversary also receives the decommitment vo_{sub} to the subgraph. Graph privacy demands now that the adversary cannot identify the original graph better than with the guessing probability. It suffices here to consider a single challenge commitment since, contrary to content privacy for graph hashes, the privacy for multiple commitments here follows from the single case via a hybrid argument.

Definition 3.3 (Graph-Privacy) *A redactable graph hashing scheme $\mathcal{H}$ is graph-private if for any probabilistic polynomial-time algorithm $\mathcal{A}$ the probability*

$$\mathrm{Prob}\left[\boldsymbol{Exp}^{GraphPriv}_{\mathcal{H},\mathcal{A}}(1^n)\right] \approx \frac{1}{2}$$

for the experiment $\boldsymbol{Exp}^{GraphPriv}_{\mathcal{H},\mathcal{A}}(1^n)$ in Figure 3 is negligibly close to $\frac{1}{2}$.

3.3 Evaluation of the Definitions of Arshad et al. [1]

In this section we evaluate the security definitions of Arshad et al. [1], after some obvious corrections for typos and stated within our terminology.

```
$\mathbf{Exp}^{\text{GraphPriv}}_{\mathcal{H},\mathcal{A}}(1^n)$
1 : $hk \leftarrow\$ \ \mathsf{HKGen}(1^n)$
2 : $(\mathtt{st}, G_{\text{sub}}, G_0, G_1) \leftarrow\$ \ \mathcal{A}(hk)$   // $G_{\text{sub}} \subseteq G_0, G_1$
3 : $b \leftarrow \{0,1\}$
4 : $(gh, vo) \leftarrow \mathsf{Hash}(hk, G_b)$
5 : $vo_{\text{sub}} \leftarrow\$ \ \mathsf{HRedact}(hk, gh, G_b, G_{\text{sub}}, vo)$
6 : $a \leftarrow \mathcal{A}(\mathtt{st}, gh, vo_{\text{sub}})$
7 : **return** 1 **if** $a = b$
```

Figure 3: Graph-Privacy Experiment for Graph Commitments. The graphs G_0, G_1 must have the same number of nodes and edges.

Collision Resistance. For the randomized hashing the definition in [1] includes a sampling algorithm $\mathcal{R}$ which outputs a random string rnd for the hash evaluation. We denote the hashing step for graph G and fixed randomness as $\mathcal{H}^{\text{rnd}}_{hk}(hk, G)$. The adversary gets to see this randomness before deciding upon the graphs. The adversary wins if it outputs distinct graphs G, G' with sub graphs $G_{\text{sub}}, G'_{\text{sub}}$, such that either the faithfully computed hash values gh, gh' of G and G' coincide, or if a genuine redaction yields a valid response for the other hash value, i.e., G_{sub} and vo_{sub} match gh', or if G'_{sub} and vo'_{sub} match gh.[2]

The main drawback of the definition in [1] is that it only covers honestly generated values for gh, vo, vo_{sub} etc. Start with a scheme which is collision-resistant according to their definition. Modify the redaction algorithm such that verification objects vo_{sub}, generated by HRedact, always carry a bit '0' in front. The modified verifier (in mode redacted) will always accept a given hash if the first bit in the input vo_{sub} is '1', and else runs the original verification procedure. This modified scheme would still be a correct redactable graph hash scheme. But an adversary could easily output vo_{sub} with a leading '1' to make the verifier accept any redacted graph and hash value, making it easy to find collisions. The security model in [1], however, would exclude such an attack by definition, because genuine values always start with '0'. The scheme would indeed preserve collision resistance in the sense of the definition, showing that the notion is not strong enough to capture admissible attacks.

We stress that in our collision-resistance experiment the adversary may choose arbitrary hash values and verification objects, such that the above successful adversary would correctly confirm that the scheme is indeed not collision-resistant according to our notion.

Privacy. Arshad et al. [1] describe two privacy experiments in their work. The first one is similar to our content-privacy experiment and says that the adversary cannot determine if it is given two hash values of either the same graph or different graphs. Yet, their experiment only considers full hash values and does not look at the privacy of redacted graphs. More precisely, the adversary is given two hashes, either of the same random graph G, of the second hash value is computer over a different random graph G', the choice depending on a secret bit b. The adversary should output a guess for b.[3]

The second privacy experiment in [1] considers a random graph G with two random sub graphs $G_{\text{sub},0}, G_{\text{sub},1}$. One first computes the hash of G and then two redacted hash values of either $G_{\text{sub},0}$, or one

[2]In our version here we corrected a flaw in the experiment in [1] where one mistakenly compares a hash value against the output of the verification algorithm.

[3]Interestingly, in the experiment the bit b is actually not used in the data communicated to the adversary, such that no adversary can do better than guessing and hence any scheme would satisfy this notion of privacy. From the description it seems reasonable to assume that our interpretation here actually complies with the intention of the authors. We also ignore the fact that the exact requirements on the distribution on the graphs are not specified.

of $G_{\text{sub},0}$ and one of $G_{\text{sub},1}$, the choice again made according to a secret bit b. It should be now be hard to decide which case has occurred.[4]

Let us first argue that the two definitions of privacy do not seem to guarantee the usual notion of privacy of the redacted parts of the original graph. Assume that we have a redactable graph hashing scheme. Modify the redaction algorithm $\mathsf{HRedact}$ now such that, on input G, G_{sub} (and potentially the hash value gh of G), it creates the verification object vo as before, but now also appends some information about original graph G in clear to vo, e.g., for sake of concreteness suppose it appends the edges and the contents of redacted nodes.

As for privacy note that, if one sees the redacted graph with the augmented verification object, then this clearly leaks information about the original graph. At the same time, if the underlying scheme satisfies the two privacy experiments before, then so does the modified scheme for some "natural" graph distributions. The reason is that the first security experiment only considers full hash values —which remain unchanged— and the sub graphs in the second experiment both start from the same original graph, such that they contain the same data in the verification object.

Consider for example distributions generating complete graphs where the content of each node is determined by an independent random string from $\{0,1\}^n$, and where the sub graphs remove a distinct node (say, the first one in some fixed order) and all edges from or to this node. Then we can formally prove privacy of redacted graphs for such distributions for our modified scheme via reduction to the privacy of the original scheme. The reduction can easily simulate the information about the redacted node by picking a random content and adding it and the edges to the verification object vo_{sub}, such that privacy according to the definitions would be preserved. Since, on the other hand, actual data is leaked, the example shows that the two privacy requirements are too weak.

Let us again emphasize that content privacy according to our definition is easy to violate by picking the well-formed distribution which augments the sub graph to a complete super graph with random content. Then putting the redacted information into the verification object, as in the scheme above, allows to distinguish different graph with overwhelming probability.

4 Constructions

In this section we describe our construction of a redactable graph hashing scheme.

4.1 Basic Construction

The basic construction is similar to the idea [1] and first hashes all nodes and edges individually. Then one can use advanced structures like iterated hash function evaluations or Merkle hash trees to combine these hash values into a shorter representation. We outline the second step in Section 4.6 but focus on the basic hashing step for now.

The hashing is based on a cryptographic hash function (for redactable graph hashing) or on a commitment (for sanitizable graph commitments). To capture all possibilities simultaneously, randomized hash functions, random oracle based solutions, and commitments, we abstractly speak of a cryptographic hash function $\mathsf{CHash} = (\mathsf{CHKGen}, \mathsf{CHash}, \mathsf{CHVf})$. This function consists of a key generation algorithm $chk \leftarrow \mathsf{CHKGen}(1^n)$, the (probabilistic) hash function $(ch, cvo) \leftarrow \mathsf{CHash}^{\mathsf{RO}}(chk, y)$, having possibly access to a random oracle $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^n$ and possibly generating some additional verification object, as

[4] The same discussion as in the first experiment about the misplaced secret bit b and the under-specified graph distributions applies here as well. We also note that Arshad et al. [1] define privacy to hold if the probability that the adversary wins the first *or* the second experiment is negligibly close to $\frac{1}{2}$. However, note that an adversary outputting a coin toss would win in at least one of the experiments with probability $\frac{3}{4}$, such that no scheme could achieve the notion in [1]. The obvious correction, which we also assume from now on, is that for both experiments the individual probability is negligibly close to $\frac{1}{2}$.

well as the verification algorithm $d \leftarrow \mathsf{CHVf}^{\mathsf{RO}}(chk, ch, cvo, y)$. Note that this also captures commitment schemes where cvo corresponds to the (initially secret) decommitment. We discuss the required security properties when presenting the concrete instantiations for the different cases.

Our construction of the redactable graph hashing scheme $\mathcal{H} = (\mathsf{HKGen}, \mathsf{Hash}, \mathsf{HRedact}, \mathsf{HVf})$ is as follows.

Key Generation $\mathsf{HKGen}(1^n)$: The key of our graph hashing scheme is given by the key of the cryptographic hash function $hk = chk \leftarrow \mathsf{CHKGen}(1^n)$.

Hashing $\mathsf{Hash}^{\mathsf{RO}}(\boldsymbol{hk}, G)$: To hash a graph $G = (V, E, \textsc{content})$ we first pick a random permutation π over $\{1, 2, \ldots, m+n\}$ for the ordered entries in V and E. Then we go through the nodes and edges. For each node $v \in V$ we compute with the cryptographic hash function the value

$$(ch_v, cvo_v) \leftarrow \mathsf{CHash}^{\mathsf{RO}}(chk, 0||v||\textsc{content}(v)).$$

For each edge $e \in E$ we compute the hash value

$$(ch_e, cvo_e) \leftarrow \mathsf{CHash}^{\mathsf{RO}}(chk, 1||e||\textsc{content}(e)).$$

In both case we assume that the node or edge identifier is represented with some fixed-length encoding (in the sum of numbers n of nodes and m of edges).

The overall hash value gh of the graph is given by the individual hash values ch_x of all elements $x \in V \cup E$ according to the random order $\textsc{order}_\pi$. Note that this in particular means that the hash value reveals the number of elements $|V| + |E|$ of the graph. The verification object vo consists of (a description of) the random permutation π and all individual verification objects cvo in the same order $\textsc{order}_\pi$. For sake of concreteness we assume that the description of π is given by the sequence $(\pi(1), \ldots, \pi(m+n))$. Then we have

$$\begin{aligned} gh &\leftarrow (ch_{\textsc{order}_\pi^{-1}(1)}, \ldots, ch_{\textsc{order}_\pi^{-1}(n+m)}) \\ vo &\leftarrow (\pi, cvo_{\textsc{order}_\pi^{-1}(1)}, \ldots, cvo_{\textsc{order}_\pi^{-1}(n+m)}). \end{aligned}$$

Redaction $\mathsf{HRedact}^{\mathsf{RO}}(\boldsymbol{hk}, \boldsymbol{gh}, G, G_{\mathbf{sub}}, \boldsymbol{vo})$: To redact a hash value gh, consisting of a sequence of hash values ch, first check that G_{sub} really is a sub graph of G and that the hash values ch for all nodes and edges in the sub graph are correct. If so, then replace all verification objects cvo in vo of nodes and edges which do *not* appear in G_{sub} by a special symbol $\perp$. The position can be easily determined with the help of the random order $\textsc{order}_\pi$. Then, also redact the description of the permutation π by creating $\pi_{\text{sub}} : V_{\text{sub}} \cup E_{\text{sub}} \to \{1, 2, \ldots, m+n\}$ which coincides with the values of π for all elements x in the sub graph:

$$\pi(\textsc{order}(x)) = \pi_{\text{sub}}(\textsc{order}_{\text{sub}}(x))$$

for the implicit order $\textsc{order}_{\text{sub}}$ for the sub graph. In particular, the description of π_{sub} consists of a sequence of $|V_{\text{sub}}| + |E_{\text{sub}}|$ distinct numbers from $\{1, 2, \ldots, m+n\}$. Let vo_{sub} be the redacted object.

Verification $\mathsf{HVf}^{\mathsf{RO}}(\boldsymbol{hk}, \boldsymbol{gh}, G, \boldsymbol{vo}, \mathsf{vfmode})$: The algorithm first checks that G has at most the same number of nodes and edges as there are entries ch in $gh = (ch_1, \ldots, ch_{m+n})$. If so, then recover the order $\textsc{order}_\pi$ from the verification object $vo = (\pi, cvo_1, \cdots cvo_{m+n})$. For each node v in G check for the $\textsc{order}_\pi(v)$-th entries $ch_{\textsc{order}_\pi(v)}$ (in gh) and $cvo_{\textsc{order}_\pi(v)}$ (in vo) that

$$cvo_{\textsc{order}_\pi(v)} \neq \perp \text{ and } \mathsf{CHVf}^{\mathsf{RO}}(chk, ch_{\textsc{order}_\pi(v)}, 0||v||\textsc{content}(v), cvo_{\textsc{order}_\pi(v)}) = 1.$$

For each edge e in G check for the $\textsc{order}_\pi(e)$-th entry $ch_{\textsc{order}_\pi(e)}$ and $cvo_{\textsc{order}_\pi(e)}$ that

$$cvo_{\textsc{order}_\pi(e)} \neq \perp \text{ and } \mathsf{CHVf}^{\mathsf{RO}}(chk, ch_{\textsc{order}_\pi(e)}, 1||e||\textsc{content}(e), cvo_{\textsc{order}_\pi(e)}) = 1.$$

Finally, for mode $\mathsf{vfmode} = \mathsf{hashed}$ also check that there is no entry $cvo = \perp$ in vo. If all these tests succeed, then output 1; else return 0.

4.2 Collision Resistance of the Basic Construction

We first argue collision resistance of the basic construction, based on the security properties of the underlying cryptographic hash function $\mathcal{CH}$. Assume that this hash function is collision-resistant in the sense that for any probabilistic polynomial-time algorithm $\mathcal{C}$ the probability that $(ch, ch', cvo, cvo', y, y') \leftarrow \mathcal{C}^{\mathsf{RO}}(hk)$ satisfies $y \neq y'$, $\mathsf{CHVf}^{\mathsf{RO}}(hk, ch, cvo, y) = \mathsf{CHVf}^{\mathsf{RO}}(hk, ch', cvo', y') = 1$ is negligible, where the probability is taken over the choice $hk \leftarrow \mathsf{CHKGen}(1^n)$ and $\mathcal{C}$'s internal coin tosses. Denoting this experiment by $\mathbf{Exp}^{\mathrm{Coll}}_{\mathcal{CH},\mathcal{C}}(1^n)$ we thus require that

$$\mathrm{Prob}\left[\mathbf{Exp}^{\mathrm{Coll}}_{\mathcal{CH},\mathcal{C}}(1^n)\right] \approx 0$$

for any probabilistic polynomial-time algorithm $\mathcal{C}$.

Theorem 4.1 *If $\mathcal{CH}$ is a collision-resistant hash function, then our redactable graph hashing scheme in Section 4.1 is collision-resistant. That is, for each probabilistic polynomial-time adversary $\mathcal{A}$ there exists a probabilistic polynomial-time adversary $\mathcal{C}$ such that*

$$\mathrm{Prob}\left[\boldsymbol{Exp}^{CR}_{\mathcal{H},\mathcal{A}}(1^n)\right] \leq \mathrm{Prob}\left[\boldsymbol{Exp}^{Coll}_{\mathcal{CH},\mathcal{C}}(1^n)\right].$$

Proof. Assume that we have an adversary $\mathcal{A}$ against the collision resistance of the graph hashing scheme. In particular, $\mathcal{A}$ on input $hk = chk$ outputs (gh, G, G', vo, vo') such that G verifies (as a full hash in mode hashed), and G' verifies in an arbitrary mode, but such that $G' \not\subseteq G$. It is straightforward to build an adversary $\mathcal{C}$ against the underlying hash function. Adversary $\mathcal{C}$ receives a key chk as input and invokes $\mathcal{A}$ on input $hk = chk$ to obtain $\mathcal{A}$'s output (gh, G, G', vo, vo'). Algorithm $\mathcal{C}$ produces its collision as described next, where $\mathcal{C}$ always succeeds in finding a collision if $\mathcal{A}$ succeeds for the graphs.

Note that G verifies in mode hashed. This means that all entries ch in gh are valid hash values of the corresponding node or edge. Next, observe that $G' = (V', E', \textsc{content}')$ is not a sub graph of $G = (V, E, \textsc{content})$. It must thus have a node $v' \in V' \setminus V$ or an edge $e' \in E' \setminus E$, or, alternatively, it must have a different content $\textsc{content}'(x) \neq \textsc{content}(x)$ for one of the elements $x \in (V' \cup E') \subseteq (V \cup E)$.

In the first case, a fresh node $v' \notin V$, verification of the redacted graph in our scheme can only succeed if the value $0||v'||\textsc{content}'(v')$ verifies under the cryptographic hash function $\mathcal{CH}$ for one of the entries. Since all entries for G are hash values of different nodes $v \neq v'$, or start with a bit '1' for edges, this can only happen if adversary $\mathcal{A}$ produces a collision.[5] This collision can be easily identified and output by adversary $\mathcal{C}$. The second case, a fresh edge $e' \notin E$, follows analogously.

The final case is that the graph G' has a different content in some element x. If the permutation π' in the verification object vo' of G' points to a different position in the hash value than for G, we immediately get a collision as in the previous cases. Else, both permutations map this element x to the same unique position. Verification can now only succeed if both (distinct) values $\xi_x||x||\textsc{content}(x) \neq \xi'_x||x||\textsc{content}'(x)$ verify under the same hash value ch, where ξ_x and ξ'_x are 0 (for a node) and 1 (for an edge). This collision is again easy to find for adversary $\mathcal{C}$. □

4.3 Content-Privacy of the Basic Construction

We next argue content-privacy of our basic construction. Recall that in the experiment the adversary chooses a sub graph G_{sub} and a well-formed distribution $\mathcal{G}$, meaning that all randomly generated graphs are super graphs of G_{sub}, have the same sets V and E, and have non-trivial entropy with respect to $\mathcal{G}$. The adversary then gets to see t hashes (and verification objects) of the same super graph, or of t independent super graphs, and should decide which is the case.

[5] Let us stress here that prepending the bit '0' or '1' excludes confusion of hashes when V' may have a non-empty intersection with E.

For the content privacy we need that the underlying hash function is t-valued perfectly one-way [10]. This means that for a non-trivial input distribution $\mathcal{Y}$ one cannot distinguish t (randomized) hash values of the same pre-image $y \leftarrow \mathcal{Y}(1^n)$ from t hashes of independent samples $y_1, \ldots, y_t \leftarrow \mathcal{Y}(1^n)$. More formally, let $(\mathtt{st}, \mathcal{Y}) \leftarrow \mathcal{D}(chk)$ and $d \leftarrow \mathcal{D}(\mathtt{st}, ch_1, vo_1, \ldots, ch_t, vo_t)$ where, depending on a random bit $b \leftarrow \{0,1\}$, either $(ch_i, vo_i) \leftarrow \mathsf{CHash}(chk, y)$ for $y \leftarrow \mathcal{Y}(1^n)$, or $(ch_i, vo_i) \leftarrow \mathsf{CHash}(chk, y_i)$ for $y_1, \ldots, y_t \leftarrow \mathcal{Y}(1^n)$. The probability, over the choice of chk, the sampling of b, y resp. $y_1, \ldots, y_t$, the hashing steps, and $\mathcal{D}$'s internal randomness, that $b = d$ should be negligible close to $\frac{1}{2}$. For any well-formed distribution $\mathcal{Y}$, i.e., which has super-logarithmic min-entropy.

For our theorem we actually need a multi-dimensional version of the above property, saying that the distribution $\mathcal{Y}$ outputs a vector $(x^1, \ldots, x^s)$ of values, where each x^i has super-logarithmic min-entropy, even when seeing the other values. Each entry is hashed individually (with fresh randomness), and the adversary gets to see either t hashes of the same vector, or of t independently sampled vectors. Denoting the experiment by $\mathbf{Exp}^{\mathrm{POW}}_{\mathcal{CH},\mathcal{D},s,t}(1^n)$ we thus require that

$$\mathrm{Prob}\left[\mathbf{Exp}^{\mathrm{POW}}_{\mathcal{CH},\mathcal{D},s,t}(1^n)\right] \approx \tfrac{1}{2}$$

for any probabilistic polynomial-time algorithm $\mathcal{D}$. We call such functions (s,t)-valued perfectly one-way hash function and argue later how to instantiate them.

Theorem 4.2 *If $\mathcal{CH}$ is a (s,t)-valued perfectly one-way hash function, then our redactable graph hashing scheme in Section 4.1 is content-private for parameter t. That is, for each probabilistic polynomial-time adversary $\mathcal{A}$ there exists a probabilistic polynomial-time adversary $\mathcal{D}$ such that*

$$\mathrm{Prob}\left[\boldsymbol{Exp}^{\mathit{ContPriv}}_{\mathcal{H},\mathcal{A},t}(1^n)\right] \leq \mathrm{Prob}\left[\boldsymbol{Exp}^{\mathit{POW}}_{\mathcal{CH},\mathcal{D},s,t}(1^n)\right]$$

where s is bounded by the number of elements $V \cup E$ in graphs generated according to distribution $\mathcal{G}$ for security parameter n.

Proof. Assume that we have an adversary $\mathcal{A}$ against the content-privacy of the graph hashing scheme. We build an adversary $\mathcal{D}$ against the multi-valued perfectly one-way hash function. Our adversary $\mathcal{D}$ against $\mathcal{CH}$ receives a hash key chk and forwards this to $\mathcal{A}$. Adversary $\mathcal{A}$ outputs $\mathtt{st}_{\mathcal{A}}, G_{\mathrm{sub}}$ and a well-formed distribution $\mathcal{G}$. Our adversary $\mathcal{D}$ generates a distribution $\mathcal{Y}$ which samples a graph G according to $\mathcal{G}$ and then outputs the values $\xi_x||x||\textsc{content}(x)$ according to the $\textsc{order}$ of elements, but only for those elements x which are not in G_{sub}. Here ξ_x is again the bit indicating whether the element is a node or an edge. Since we assume that $\mathcal{G}$ is well-formed, each content value $\textsc{content}(x)$ has super-logarithmic min-entropy, even given the other values, such that the distribution $\mathcal{Y}$ is admissible.

Our adversary $\mathcal{D}$ augments each of the t vectors of values by locally computing the values $(ch_x, cvo_x) \leftarrow \mathsf{CHash}(chk, \xi_x||x||\textsc{content}(x))$ for all x in the sub graph. For each of these augmented t vectors of now $m+n$ elements our adversary picks a random permutation π_i and creates the hash value and verification objects according to the scheme's description. Adversary $\mathcal{D}$ hands over $\mathtt{st}_{\mathcal{A}}$ and all hashes gh_i, vo_i for $i = 1, 2, \ldots, t$ to adversary $\mathcal{A}$. When $\mathcal{A}$ returns a bit a, then our algorithm $\mathcal{D}$, too, returns this bit.

For the analysis note that if the challenger in the perfectly one-wayness experiment creates the t hash vectors according to a single sample from $\mathcal{Y}$, then this perfectly mimics the content-privacy game where the same graph is used. Analogously, if we obtain independent hashes, then this perfectly mimics the content-privacy experiment in this case. This is true since all possible super graphs have the same structure V and E, such that the elements of the sub graphs appear at the same ordered positions, in both cases. Therefore, the actual random positions of the elements in the sub graph together with the information about π do not reveal anything about the super graph.[6] $\square$

[6]This would not be true if the distribution $\mathcal{G}$ could generate different structures, because the adversary could then identify

4.4 Graph-Privacy of the Basic Construction

Graph privacy says that the adversary, picking G_{sub} and super graphs G_0, G_1 of the same number of elements, cannot distinguish redacted hash values (and the redacted verification object) when we start with either G_0 or with G_1 and redact to G_{sub}. Note that the adversary does not get to see the original verification object.

For the security we need the hash function to be a commitment scheme, having the same collision resistance property as before (also called binding property in the context of commitments) and also the hiding property. The latter says that any adversary $\mathcal{H}$, upon input *chk* determining two inputs y_0, y_1, and receiving the hash value *chk* of $(chk, cvo) \leftarrow \mathsf{CHash}(chk, y_b)$ for random bit $b \leftarrow \{0,1\}$ but not the verification object (aka. decommitment), cannot predict b significantly better than by guessing. More formally, let $(\mathtt{st}, y_0, y_1) \leftarrow \mathcal{H}(chk)$ and $h \leftarrow \mathcal{H}(\mathtt{st}, ch)$ where, depending on a random bit $b \leftarrow \{0,1\}$, either $(ch, cvo) \leftarrow \mathsf{CHash}(chk, y_b)$. Denoting the experiment by $\mathbf{Exp}^{\text{Hide}}_{\mathcal{CH},\mathcal{H}}(1^n)$ we thus require for a hiding commitment scheme that

$$\text{Prob}\left[\mathbf{Exp}^{\text{Hide}}_{\mathcal{CH},\mathcal{H}}(1^n)\right] \approx \tfrac{1}{2}$$

for any probabilistic polynomial-time algorithm $\mathcal{H}$, where the probability is defined over the choice of *chk*, the bit b, and $\mathcal{H}$'s internal randomness.

We note that, via a standard hybrid argument, it is also hard to distinguish sequences of commitments, either all for "left" values $x_{0,1}, \ldots, x_{0,t}$ or all for "right" values $x_{1,1}, \ldots, x_{1,t}$, both sequences chosen by the adversary. The advantage (over the guessing probability $\frac{1}{2}$) in distinguishing such sequences versus the advantage for a single commitment, is at most a factor t larger. In our setting, $t = m + n$ will be polynomial such that the advantage is still negligible.

Theorem 4.3 *If $\mathcal{CH}$ is a hiding commitment scheme, then our redactable graph hashing scheme in Section 4.1 is graph-private. That is, for each probabilistic polynomial-time adversary $\mathcal{A}$ there exists a probabilistic polynomial-time adversary $\mathcal{D}$ such that*

$$\text{Prob}\left[\boldsymbol{Exp}^{\mathit{GraphPriv}}_{\mathcal{H},\mathcal{A}}(1^n)\right] \leq \frac{1}{2} + (n+m) \cdot \left|\text{Prob}\left[\boldsymbol{Exp}^{\mathit{Hide}}_{\mathcal{CH},\mathcal{H}}(1^n)\right] - \frac{1}{2}\right|.$$

Proof. Consider again an adversary $\mathcal{A}$ against the graph privacy, outputting G_{sub}, G_0, G_1 after having seen *hk*, and being able to predict the bit b from the redacted hash value for graph G_b. From $\mathcal{A}$ we construct an adversary $\mathcal{H}$ against the hiding property of the commitment scheme (where we assume that $\mathcal{H}$ receives a sequence of at most $m + n$ commitments).

Algorithm $\mathcal{H}$ initially receives the commitment key *chk* and forwards it to $\mathcal{A}$. It then gets graphs $G_{\text{sub}} \subseteq G_0, G_1$ (and some state information $\mathtt{st}_{\mathcal{A}}$). Algorithm $\mathcal{H}$ picks a random order $\textsc{order}_\pi$ for the two graphs G_0, G_1 (with the same number of elements). To prepare the simulated commitment of the graph for $\mathcal{A}$, for each element x in G_{sub} algorithm $\mathcal{H}$ computes $(ch_x, cvo_x) \leftarrow \mathsf{CHash}(chk, \xi_x || x || \textsc{content}(x))$ locally, where the bit ξ_x is 0 or 1, depending on whether x is a node or an edge. For all the other elements x in graph G_0 it prepares the sequence of values $\xi || x || \textsc{content}(x)$, leaving out elements in G_{sub}. Denote this sequence of $t = m + n - |V_{\text{sub}}| - |E_{\text{sub}}|$ entries as X_0. The order here is irrelevant. Prepare the (equal-length) sequence X_1 for G_1 analogously. Algorithm $\mathcal{H}$ outputs X_0, X_1 for the commitment challenge.

When $\mathcal{H}$ receives a sequence $(ch_1, \ldots, ch_t)$ as a challenge, it creates the (redacted) value in the simulation of $\mathcal{A}$ as follows. First it permutes the values $ch_1, \ldots, ch_t$ randomly. Then it "mixes in" the locally computed values for elements in G_{sub} at random positions. Collect this mapping of the elements in the sub graph in a random mapping π_{sub}. This yields a hash value *gh*. For the verification object of the redacted graph

via π where the elements of the sub graph appear in the order of the super graph. These positions usually vary for t independent samples.

compute a vector $(\bot, \ldots, \bot)$ and insert the locally computed objects *cvo* for elements in G_{sub} at the right positions according to π_{sub}, such that together with π_{sub} one obtains vo_{sub}.

Continue $\mathcal{A}$'s execution on $\texttt{st}_{\mathcal{A}}, gh, vo_{\text{sub}}$ to obtain a bit a. Return a as the prediction for the secret bit b in the commitment game, too.

For the analysis note that if the commitment challenger commits to the sequence X_0, i.e., $b = 0$, then we obtain a perfect simulation of a sanitized commitment of graph G_0. The reason is that the initial hash computation according to the graph-privacy experiment would randomly permute the entries in the hash value and place the elements of the sub graph at random positions. The redacted verification object would thus, besides the decommitments of such elements, also contain a random position for each element. This is identical to our construction.

The argument for $b = 1$, when receiving commitments to X_1, is identical and shows that $\mathcal{A}$'s view is perfectly simulated as in the game where G_1 is redacted. Hence, $\mathcal{A}$' prediction capabilities from the graph privacy experiment immediately transfer to $\mathcal{H}$'s attack against the commitment scheme, we only lose the factor $m + n$ due to the fact that we use multiple commitments. $\square$

4.5 Instantiations

The above general construction does not specify the underlying cryptographic hash function. We discuss here potential instantiations for the hash function.

Collision-Resistance. If we are purely interested in a collision-resistant redactable graph hashing, then any common collision-resistant hash function will work. Here, a deterministic collision-resistant function suffices, where the verification object is empty and verification is performed by re-computing the hash value and comparing it to the given value.

Content-Privacy. If we are interested in content-privacy then it is tempting to take a t-valued perfectly one-way hash function. Such hash functions can be built from regular collision-resistant hash functions [10]. Yet, we need a multi-dimension version of it, and in general it is not known if a t-values function has this property. The reason is that the input may have sufficient individual min-entropy, but it may be highly correlated, such that a common hybrid argument may thus not be immediately applicable.

A random oracle based solution works, though. For this define $\mathsf{CHash}^{\mathsf{RO}}(hk, y)$ to return the random oracle value $\mathsf{RO}(r||y)$ together with the fixed-length randomness $r \in \{0,1\}^n$. Then the function is collision-resistant, since a random oracle has this property. Also, the individual super-logarithmic min-entropy ensures that the adversary most likely does not query any of the unknown pre-images, except with negligible probability. But then the hash values (of the same y or distinct $y_1, \ldots, y_t$) are perfectly indistinguishable for the adversary, for any polynomial number of samples.

Graph-Privacy. For graph-privacy any (non-interactive) commitment scheme processing arbitrary input lengths suffices. As a concrete example one can use the one based on collision-resistant hashing [11, 17]. Alternatively, one can use a random oracle based solution via $\mathsf{RO}(r||y)$, where $r \in \{0,1\}^n$ together with y serves as the verification object/decommitment.

4.6 Deploying Hash Trees

Our basic solution is quite expensive in terms of the length of hash values. The hash values are linear in $|V| + |E|$. We note that one can build a Merkle hash tree [21] on top to shrink hash values to a single element. That is, starting from the given individual hash values, one progressively hashes together two values from the previous stage in a tree structure, till one reaches a single root node.

To avoid collisions with the hashing of nodes an edges we propose domain separation and let each hash computation in the tree start with '2', where it is understood that now the numbers '0' and '1' in the basic construction are represented with two bits. Furthermore, the construction requires that one can actually re-compute a hash value *ch* from the input and the verification object. This is usually the case for hash functions with public randomness.

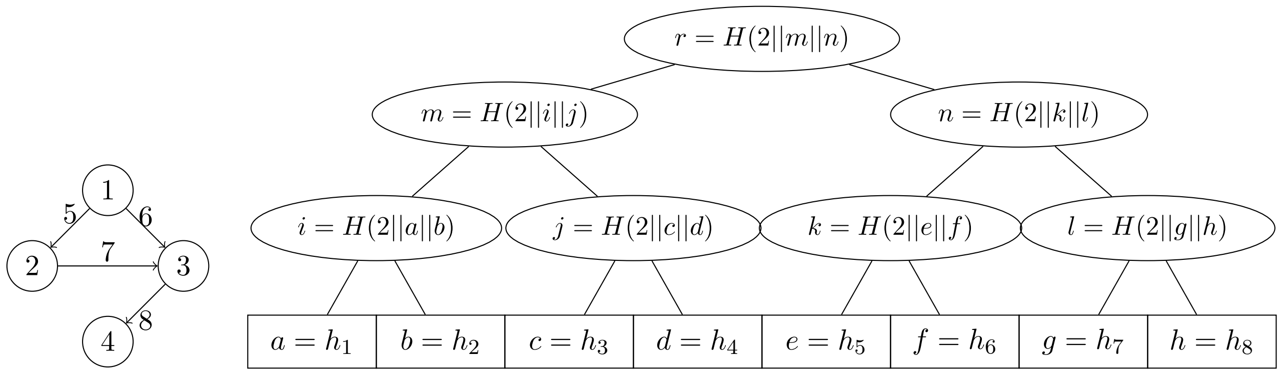

Figure 4: Example generation of the Merkle hash tree for the graph on the left hand side. The numbers in the graph correspond to the random order $\textsc{order}_\pi$. In the hash tree we have $(h_i, vo_i) \leftarrow \mathsf{CHash}(chk, \xi||\textsc{order}_\pi^{-1}(i)||\textsc{content}(\textsc{order}_\pi^{-1}(i)))$, and the hash function H may also be taken to be $\mathsf{CHash}(chk, \cdot)$. The graph hash function's output is the root value r of the tree. The verification object of the graph hash is given by π and all the verification objects $vo_1, \ldots, vo_8$ of the hash values at the leaves.

The verification object for an original hash value of an element would then need to include the verification object as before, but also all intermediate hash values in the tree on the path to the root for sub trees which are redacted. The idea is displayed in Figures 4 and 5 for a simple example. We note that the size of the verification object cannot increase by more than the size of the hash value in our basic construction, since we include at most the original hash values of all leaves, but potentially even less for large redacted sub trees.

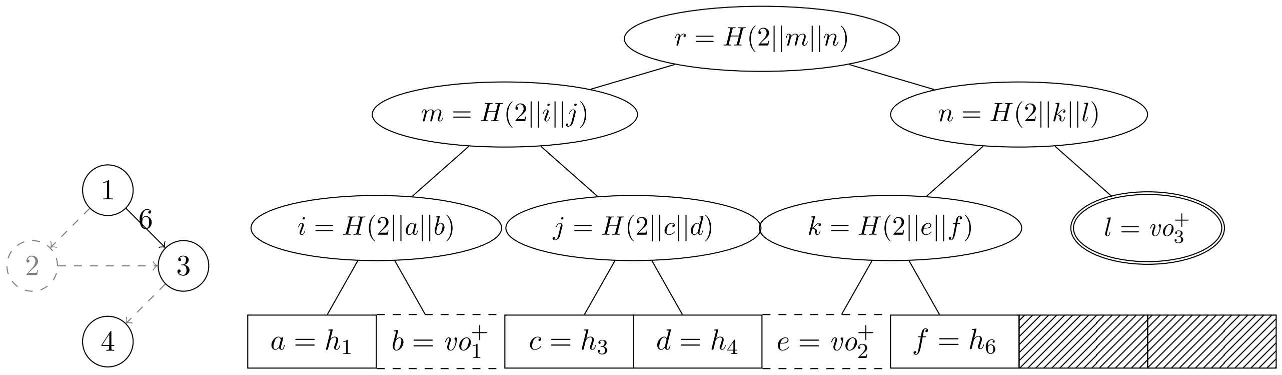

Figure 5: Redaction example for the graph in Figure 4. The redacted graph consists of the elements with random order numbers $1, 3, 4$, and 6. The original verification object is given by $vo_{\text{sub}} = (\pi_{\text{sub}}, vo_1, \perp, vo_3, vo_4, \perp, vo_6, \perp, \perp)$ but is augmented by the additional objects $vo_1^+ = h_2$, $vo_2^+ = h_5$ (with dotted lines) and $vo_3^+ = H(2||h_7||h_8)$ (with double line).

5 Conclusion

We propose strong security notions for collision resistance and privacy for redactable graph hashing (or sanitizable graph commitments), as well as constructions achieving the notions. One of the most interesting aspects to be investigated is to improve the efficiency of such solutions. Our constructions produce quite large hash values, albeit this can be alleviated by applying common strategies like Merkle hash trees. Getting even shorter solutions is an interesting research topic. Alternatively, one may be able to show lower bounds for the size of hash values and verification objects for general redaction schemes.

Then, our redaction is quite general in the sense that one can also, say, redact edges for a graph. Suppose that nodes correspond to persons in a social network and the edges represent the contacts. Then one may be interested in seeing only the data for some persons/nodes, but such that all edges for such nodes are preserved, e.g., to accurately count the number of contacts. It would be interesting to derive schemes with specific redaction procedures, such as the projection onto nodes (with edges being preserved). While such procedures can be easily implemented with our general approach, they pose additional security requirements, e.g., in the example it must be guaranteed that edges for nodes cannot be dropped. Interestingly, and connecting to the first point, one may be even able to provide more efficient constructions for such specialized redaction schemes.

Acknowledgments

We thank the anonymous reviewers for comments. This work has been co-funded by the German Research Foundation DFG as part of project P2 within the CRC 1119 CROSSING and the SPP 1736 grant Fi 940/5-1.

References

[1] Arshad, M.U., Kundu, A., Bertino, E., Madhavan, K., Ghafoor, A.: Security of graph data: Hashing schemes and definitions. In: Proceedings of the 4th ACM Conference on Data and Application Security and Privacy. pp. 223–234. CODASPY '14, ACM, New York, NY, USA (2014), `http://doi.acm.org/10.1145/2557547.2557564` (Cited on pages 1, 2, 3, 4, 5, 6, 7, 8, and 9.)

[2] Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3679, pp. 159–177. Springer (2005) (Cited on page 1.)

[3] Bauer, D., Blough, D.M., Mohan, A.: Redactable signatures on data with dependencies and their application to personal health records. In: Proceedings of the 2009 ACM Workshop on Privacy in the Electronic Society, WPES 2009, Chicago, Illinois, USA, November 9, 2009. pp. 91–100. ACM (2009) (Cited on page 1.)

[4] Brown, J., Ahamad, M., Ahmed, M., Blough, D.M., Kurc, T., Post, A., Saltz, J.: Redactable and auditable data access for bioinformatics research pp. 21–25 (2013) (Cited on page 1.)

[5] Brown, J., Blough, D.M.: Verifiable and redactable medical documents. In: AMIA 2012, American Medical Informatics Association Annual Symposium, Chicago, Illinois, USA, November 3-7, 2012. AMIA (2012) (Cited on page 1.)

[6] Brzuska, C., Busch, H., Dagdelen, Ö., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., Schröder, D.: Redactable signatures for tree-structured data: Definitions and constructions. In: Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6123, pp. 87–104 (2010) (Cited on page 3.)

[7] Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5443, pp. 317–336. Springer (2009) (Cited on page 2.)

[8] Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of sanitizable signatures. In: Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6056, pp. 444–461. Springer (2010) (Cited on page 2.)

[9] Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO'97. Lecture Notes in Computer Science, vol. 1294, pp. 455–469. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 17–21, 1997) (Cited on pages 3 and 6.)

[10] Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions (preliminary version). In: 30th Annual ACM Symposium on Theory of Computing. pp. 131–140. ACM Press, Dallas, Texas, USA (May 23–26, 1998) (Cited on pages 3, 6, 12, and 14.)

[11] Damgård, I., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. Journal of Cryptology 10(3), 163–194 (1997) (Cited on page 14.)

[12] Devanbu, P.T., Gertz, M., Kwong, A., Martel, C.U., Nuckolls, G., Stubblebine, S.G.: Flexible authentication of XML documents. In: CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001. pp. 136–145. ACM (2001) (Cited on pages 2 and 3.)

[13] Devanbu, P.T., Gertz, M., Martel, C.U., Stubblebine, S.G.: Authentic third-party data publication. In: Data and Application Security, Development and Directions, IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security, Schoorl, The Netherlands, August 21-23, 2000. IFIP Conference Proceedings, vol. 201, pp. 101–112. Kluwer (2000) (Cited on pages 2 and 3.)

[14] Devanbu, P.T., Gertz, M., Martel, C.U., Stubblebine, S.G.: Authentic data publication over the internet. Journal of Computer Security 11(3), 291–314 (2003) (Cited on pages 2 and 3.)

[15] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing. pp. 169–178. ACM Press, Bethesda, Maryland, USA (May 31 – Jun 2, 2009) (Cited on page 1.)

[16] Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Efficient authenticated data structures for graph connectivity and geometric search problems. Algorithmica 60(3), 505–552 (2011) (Cited on page 3.)

[17] Halevi, S., Micali, S.: Practical and provably-secure commitment schemes from collision-free hashing. In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO'96. Lecture Notes in Computer Science, vol. 1109, pp. 201–215. Springer, Berlin, Germany, Santa Barbara, CA, USA (Aug 18–22, 1996) (Cited on page 14.)

[18] Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic signature schemes. In: Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2271, pp. 244–262. Springer (2002) (Cited on page 1.)

[19] Kundu, A., Bertino, E.: Structural signatures for tree data structures. PVLDB 1(1), 138–150 (2008) (Cited on page 3.)

[20] Martel, C.U., Nuckolls, G., Devanbu, P.T., Gertz, M., Kwong, A., Stubblebine, S.G.: A general model for authenticated data structures. Algorithmica 39(1), 21–41 (2004) (Cited on pages 1, 2, 3, and 4.)

[21] Merkle, R.C.: A certified digital signature. In: Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer (1989) (Cited on pages 3 and 14.)

[22] Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., de Meer, H.: On structural signatures for tree data structures. In: Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7341, pp. 171–187. Springer (2012) (Cited on page 3.)

[23] Samelin, K., Pöhls, H.C., Bilzhause, A., Posegga, J., de Meer, H.: Redactable signatures for independent removal of structure and content. In: Information Security Practice and Experience - 8th International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7232, pp. 17–33. Springer (2012) (Cited on page 3.)

[24] Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2288, pp. 285–304. Springer (2001) (Cited on page 1.)

[25] Wu, Z.Y., Hsueh, C., Tsai, C., Lai, F., Lee, H., Chung, Y.: Redactable signatures for signed CDA documents. J. Medical Systems 36(3), 1795–1808 (2012) (Cited on page 1.)