Verifiable Document Redacting

Hervé Chabanne[1,2], Rodolphe Hugel[1], and Julien Keuffer[1,3]

[1] Morpho, Issy-les-Moulineaux, France
[2] Telecom ParisTech, Paris, France
[3] Eurecom, Biot, France
{herve.chabanne,rodolphe.hugel,julien.keuffer}@morpho.com

Abstract. In 2016, Naveh and Tromer introduced PhotoProof, a novel approach to image authentication based on cryptographic proofs. We here show how to simplify PhotoProof to get a protocol closely related to redactable signature schemes. From an authenticated breeder document, we only keep the necessary fields to prove what its owner wants to assert and black out all the others to remove sensitive data from the document. We efficiently instantiate our scheme and give implementation results that show its practicality.

Keywords: data privacy, zk-SNARK, redactable signatures

1 Introduction

Motivation. People are frequently asked for information such as their place of residence, a source of income or a proof of employment in order to get e.g. a traveling visa or an identity card. They can provide a document, called a *breeder document*, which will be accepted as a proof as long as the document provider is trusted by the service which needs the paper. Nevertheless, these documents might contain private information that the owner does not want to share with the service provider asking for a justification. The problem addressed in this paper is to determine whether it is possible to keep sensitive information private on a document while giving a third party assurance that the redacted document was built from an authentic one.
To illustrate the relevance of the latter problem, we below give some examples where a document contains private information useless for the required justification:

- giving a pay stub to justify employment indeed gives the name and address of the employer but also reveals a sensitive and useless information for this goal, namely the salary amount,
- someone can prove he has earnings by providing the balance of his bank statement (in order to get a visa for instance) but the detail of all the transactions written in the statement does not concern the entity needing a revenue justification,

– in some countries, for the issuance of documents like driver license or identity card, an individual has to prove his place of residence with a bill (e.g. an electricity bill) where his name is written. However, the bill can also mention the name of the partner, which has no connection with the original request.

Even if removing the sensitive information from the document looks as a natural and efficient solution to our problematic, service providers fear fraudulent document forgery and often ask to bring the original document.

In this paper, we argue that documents digitization opens the possibility to use cryptographic techniques such as signature to guarantee integrity and authenticity of the document issued by the trusted provider. Still, a problem remains: if the user makes redaction on a signed document, the signature cannot be verified with the new modified document. The client could ask the document issuer to edit a new redacted and signed version of the original document but this reveals which information is sensitive and thus is a privacy loss.

Our contribution. We propose a protocol to issue a redacted document from an original and authenticated document. Our protocol involves three parties: the issuer of the original document, the client that wants to redact the document and the document user who makes a request to the client. The protocol gives strong guarantees that nothing has been modified from the original document except the redacted parts. Moreover it links the redacted document to the original one while keeping the original one private. It should also be noticed that the issuer of the document has minimal work to do: after generating a document, he only has to compute a hash value and a signature. No further work is needed during the redaction or the verification of the document.

The main tool for building this protocol is verifiable computation [18]. In verifiable computation, a verifier delegates a computation to an untrusted prover. The prover sends back the result of the computation and a proof that the computation is correct. Several implementations of verifiable computation have been proposed [26, 7, 12, 15]. Among these schemes, we need the one having two additional properties: being *non-interactive* and providing zero-knowledge proofs for inputs supplied by the prover. Basically, our protocol is the following: the issuer first generates a document, signs a hash computed from the original document and sends the document, the hash and its signature to the client. The client then redacts some parts of the document and computes a non-interactive zero-knowledge proof, proving that only the redacted parts have been modified from the original document. The signature of the redacted document consists of the original signature and the proof. This signature has constant size and thus does not depend on the proportion of the document that has been redacted. The zero-knowledge property of the proof ensures that no information about the original document is contained in the proof. The client can then send the redacted document and its signature to the document user, along with some other elements needed to verify the proof. If the proof is correct, the document user can accept the redacted document with confidence. We stress that the document is publicly verifiable: the scheme produces verification keys and anyone with access to these

keys can verify the validity of the proof. Moreover since the proof has a constant short size, the verification is quick.

Related works. In France, the most recent proposition to secure breeder document is called 2D-DOC [1]. It is a protocol to secure physical breeder document such as electricity bill, bank statement or phone bill. The most relevant information of the document are gathered and form a blob that is digitally signed. The blob and its signature are represented as a 2D bar-code and printed on the document.This guarantees the authenticity and integrity of the document. However, if the document is redacted the signature is no longer valid with the information left. Moreover, since the 2D bar-code contains the most relevant information of the document, private data appear on the bar-code and redacting the bar-code destroys the authenticity proof of the document.

Photoproof [25] is a recent protocol enabling the authentication of images that have been modified from an original one as long as the transformations belong to a well defined set. It builds on the notion of *proof carrying data* (PCD) [14] which are data along with a proof of some property satisfied by the data. PCD enable a data to be sequentially modified, the proof containing a proof of the current property and also a proof that all the previous data modifications have satisfied the required properties. PCD can be instantiated but the computational overhead for the prover is consequent: for example in Photoproof [25], limiting the set of transformation to cropping, rotating, transposing, bit flipping and modifying the brightness of the image, the authors report 300 seconds to build a proof for a 128×128 (pixels) image. The size of the public key used to build the proof is 2 GB; in contrast the verification is less than half a second long. So, even if the requirement of integrity and of confidentiality are satisfied, there is a need to simplify the above scheme in order to reach some efficiency and to be able to deal with larger images. Indeed, an A4 format bill scanned at 100 dpi produces a 1169×827 image. Our scheme also enables image authentication, but since we only allow redaction, we obtain much better proving time. Our scheme can therefore more easily scale on image size. See Sect. 4.2 for implementation results.

Redactable signatures are strongly related to our proposal. A redactable signature allows a party to remove parts of a signed document and to update the signature without possession of the signer's secret key. Moreover, the validation of the updated signature is still possible with the signer's public key. Redactable signatures have been independantly introduced by [29] and [23]; there has been a large body of work since, e.g. [13, 28, 17]. Our proposal shares some security goals with redactable signatures such as privacy of the redacted content and unforgeability of the signature. A notable difference is that everyone can redact a document in redactable signatures schemes while in our protocol only the owner of the document can perform redaction. Indeed some private inputs of the proof computed by the redactor cannot be supplied unless being in possession of both the original document and some value used to compute the hash. Our protocol enables redacting an image, a use case for which the existing redactable schemes

would be impractical due to the length of the obtained signature or the time to generate the signature. Indeed in redactable signature schemes the length of the signature depends on the number of message blocks n and has at best a length of $\mathcal{O}(n)$. In the redaction of an image each pixel can be potentially redacted and therefore a block for an image to redact is a pixel. In contrast, our redacted signature has constant size. We finally note that our scheme cannot satisfy the transparency property as defined in [13], which states that it should be unfeasible to decide whether a signature directly comes from the signer or has been generated after some redaction. Indeed, we give places where redaction happened to the verifier and thus transparency cannot be reached. This fits however to our use case since the redacted document is given to the verifier and redacted places are thus visible to the verifier.

Organization of the paper. We define the protocol syntax and its security in Sect. 2. We give background on verifiable computation in Sect. 3. We thus instantiate our protocol in the case of document represented as images and give experimental results in Sect. 4.

2 Our protocol for redacting documents

2.1 High level description

Let the *document issuer* (DI), the *client* (CL) and the *service provider* (SP) be the three parties involved in the scheme. The document issuer first generates a document $\mathcal{D}$, computes a hash value C from $\mathcal{D}$ and a random value r. DI then signs the hash value to authenticate it and sends the client $\mathcal{D}, C, r$ and the signature of C. To give the possibility to redact the document to the client while keeping a link with the original document, we use a verifiable computation scheme to produce a proof of the statement below. In the statement, MOD is a set describing all the redacted places of the document $\mathcal{D}$. In our motivating example, MOD would be the coordinates of all the pixels of the image that are turned black.

There exists a document $\mathcal{D}$ and a set of coordinates MOD such that the redacted document $\mathcal{D}_{red}$ only differs from $\mathcal{D}$ in places defined by the set MOD.

In the proof, the original document $\mathcal{D}$ stays private using a property of verifiable computing schemes: the prover can supply a private input in the computation and build a zero-knowledge proof of the computation. The verifier thus cannot infer information about the prover's input by examining the proof. To ensure that the proof has been built with the original document, a hash computed from the original document is added to the computation. Since this hash will be sent to the verifier of the redacted document it cannot be only the hash of the document, otherwise this would give an oracle for the verifier to test the redacted parts of the document. This is why the random value r is computed by the document issuer and concatenated to the document before the hash computation. Using the same notations, the statement to be proved now becomes:

There exists a document $\mathcal{D}$ and a value r such that the hash of $\mathcal{D} \parallel r$ equals C and such that $\mathcal{D}_{red}$ only differs from $\mathcal{D}$ in places defined by the set MOD.

Denoting by π the proof, the client thus passes $\pi, \mathcal{D}_{red}, MOD, C$ and its signature σ to SP. The service provider first verifies that the signature σ of C is correct to be sure that the hash of the original document is authentic. He then uses $C, \mathcal{D}_{red}$ and MOD to verify the proof π. We stress that π ties the hash value computed and authenticated by the document issuer to the original document because it proves (in zero-knowledge) that this document, concatenated with the value r hashes into C. Thus, the correct verification of the signature of C and of the proof π guarantees that the original document is authentic. In the next section, we give a more formal description of the scheme.

2.2 The verifiable document redacting protocol

In this section we define the syntax and the security of our scheme. As it was mentioned in the introduction, the security goals of our scheme are close to the redactable signatures goals [29].

Protocol syntax. Let $(\texttt{Gen}, \texttt{Sign}, \texttt{Ver})$ be a signature scheme [11], H be a hash function and let the triple of algorithms $(\texttt{Setup}, \texttt{Prove}, \texttt{Verify})$ be a zk-SNARK [7]. See Sect. 3 for details on the zk-SNARK algorithms.
The protocol participants are the Document Issuer (DI), the Client (CL) and the Service Provider (SP). Let $M = (m_1, \ldots, m_n)$ be a message composed of n sub-messages. We use a special symbol # to denote the redaction of a sub-message. When a message M is redacted, the resulting message is denoted $M_{red} = (m_1^{red}, \ldots, m_n^{red})$. Our verifiable document redacting (**VDR**) scheme is a tuple of four polynomial time algorithms:

$\texttt{KeyGen}(1^\lambda, \mathcal{F})$: this probabilistic algorithm takes a security parameter λ and runs the $\texttt{Gen}$ algorithm to output a secret/public signing key pair (SK, PK). It then takes λ and an arithmetic circuit over a finite field $\mathbb{F}_p$, runs the $\texttt{Setup}$ algorithm and outputs a pair of public proving and verification keys $(EK_\mathcal{F}, VK_\mathcal{F})$ for the circuit $\mathcal{F}$.

$\texttt{Authent}(M, SK)$: this probabilistic algorithm, run by DI, takes a document M, a secret signing key SK and computes:
- $r \xleftarrow{\$} \{0,1\}^{128}$
- $C \leftarrow H(M \parallel r)$
- $\sigma \leftarrow \texttt{Sign}(C, SK)$

Output: (C, r, σ)

$\texttt{Redact}(M, C, r, \sigma, EK_\mathcal{F})$: this probabilistic algorithm, run by CL, takes a document M, the output (C, r, σ) computed by $\texttt{Authent}$ and the evaluation key $EK_\mathcal{F}$ and computes:
- $d \leftarrow \texttt{Ver}(C, \sigma, PK)$
- If $d = 0$, then abort.
- Else:

- define the set MOD of the redacted sub-messages, MOD is a subset of $\{1, \ldots, n\}$,
- define M_{red} such that: $\forall i \in MOD, M_{red}(i) = \#$
- $\pi \leftarrow \texttt{Prove}(([M, r], C, M_{red}, MOD), EK_{\mathcal{F}})$, where the value between brackets, namely the original document and the randomness used to compute C, are privately supplied by the prover and the circuit $\mathcal{F}$ used in `Prove` is built to verify the following statement:

$$\begin{cases} \exists M, r \text{ such that} H(M \parallel r) = C \\ \forall j \in MOD, m_j = \# \\ \forall j \notin MOD, m_j = m_j^{red} \end{cases}$$

Output: $(M_{red}, MOD, C, \sigma, \pi)$ – the signature of M_{red} is the pair (σ, π).

`DocVerif`$(M_{red}, MOD, \sigma, \pi, VK_{\mathcal{F}}, PK)$: this deterministic algorithm, run by SP, takes a redacted document M_{red}, a set of redacted sub-messages index MOD, a signature σ, a proof π and the signing public key and the verification key. It outputs a bit d such that:

- $d \leftarrow \texttt{Ver}(C, \sigma, PK)$
- $d \leftarrow d \times \texttt{Verify}(\pi, (M_{red}, C, MOD), VK_{\mathcal{F}})$

Protocol security. We now define the security goals of our scheme, adapting security notions defined in [13]. Our first goal is to reach privacy of the redacted document, informally meaning that no PPT adversary only in possession of the redacted message and its proof can recover information about the redacted parts of the message. Our second goal is unforgeability of the proof: a PPT adversary not being in possession of the original message cannot create a redacted document and a proof that will be accepted by the verifier. We formalize these goals below.

Privacy : a **VDR** scheme (`KeyGen`, `Authent`, `Redact`, `DocVerif`) is private if for all PPT adversaries $\mathcal{A}$, the probability that the experiment `Leak` evaluates to 1 is negligibly close to $\frac{1}{2}$.

The `Leak` experiment:

- $b \leftarrow \{0, 1\}$
- $(M^0, M^1, i) \leftarrow \mathcal{A}$
  with (M^0, M^1, i) such that $\forall j \neq i,\ M_j^0 = M_j^1$ and $M_i^0 \neq M_i^1$
- $(M_{red}^b, C_b, \sigma_b, \pi_b) \leftarrow \mathcal{O}^{\texttt{Auth/Redact}}$
- $b^\star \leftarrow \mathcal{A}(PK, EK_{\mathcal{F}}, VK_{\mathcal{F}}, M_{red}^b, C_b, \sigma_b, \pi_b)$
- Return 1 if $b^\star = b$

The adversary's advantage is defined as: $\mathsf{Adv}_{\mathsf{Leak}}^{\mathcal{A}} = \left|\Pr[\mathsf{LeakExp} = 1] - \frac{1}{2}\right|$
A **VDR** scheme is private if $\mathsf{Adv}_{\mathsf{Leak}}^{\mathcal{A}}$ is negligible for all PPT adversaries.

Unforgeability : a **VDR** scheme (`KeyGen`, `Authent`, `Redact`, `DocVerif`) is unforgeable if for all PPT adversaries $\mathcal{A}$, the probability that the experiment `Forge` evaluates to 1 is negligible.

The $\mathsf{Forge}(\lambda)$ experiment:
- $(SK, PK, EK_{\mathcal{F}}, VK_{\mathcal{F}}) \leftarrow \mathtt{KeyGen}(\lambda)$
- For $i = 1, \ldots, q$: $(M^i_{red}, MOD^i, \sigma^i, \pi^i) \leftarrow \mathcal{O}^{\mathtt{Auth/Redact}}$
- $(M_{red}, MOD, \sigma, \pi) \leftarrow \mathcal{A}$
- Return 1 if:
  - $\mathtt{DocVerif}(M_{red}, MOD, \sigma, \pi, VK_{\mathcal{F}}, PK) = 1$ and
  - $(M_{red}, MOD, \sigma, \pi) \neq (M^i_{red}, MOD^i, \sigma^i, \pi^i), \forall i \in \{1, \ldots, q\}$.

We define the advantage of the adversary as: $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Forge}} = |\Pr[\mathsf{ForgeExp} = 1]|$ The **VDR** scheme is unforgeable if $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Forge}}$ is negligible for all PPT adversaries.

Definition 1. *A **VDR** scheme is secure if it is private and unforgeable as defined above.*

Theorem 1. *If the signature scheme is existentially unforgeable under chosen message attack (EUF-CMA), the verifiable computing scheme is secure and the hash function is such that $H(., r)$ is a secure PRF then the **VDR** scheme is secure.*

Proof. The proof is detailed in Appendix A.

3 Verifiable Computation

With the advent of cloud computing, efficient schemes for delegation of computation have been proposed [20, 22], building on the PCP theorem [4]. Despite the improvements made, these schemes were lacking either expressiveness (only a restricted class of computation could be delegated) or concrete efficiency (constants too high in the asymptotics). Few practical-oriented constructions have been proposed by Groth [21] or Setty et al. [27] but the breakthrough of Gennaro et al. [18] really opened the way to near practical and general purpose verifiable computation schemes. Gennaro et al. introduced quadratic arithmetic programs (QAPs), an efficient way of encoding the arithmetic circuit satisfiability problem. Parno et al. [26] embedded QAPs into a bilinear group, producing a Succinct Non-interactive ARGument (SNARG) that can be turned into a zero-knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARK) with almost no additional costs. The protocol, called Pinocchio, is also publicly verifiable: public evaluation and verification keys are computed from the QAPs and anyone with access to the verification key can validate the proof. Note that in order to have an efficient verifier, SNARKs built on QAPs use an expensive pre-processing phase where evaluation and verification keys are computed, enabling to produce a constant size proof and to get a constant verification time.

There exists several zk-SNARK implemented systems, all building on QAPs. They compile a program written with a high-level language into a circuit, turning the latter into a QAP and then applying a cryptographic machinery to get a

SNARK [30, 7, 26, 16]. These systems make different trade-offs between efficiency for the prover and expressivity of the computations to be verified, comparisons can be found in the survey [31].

In the following sections, we sketch the QAP construction and the Pinocchio verifiable computing protocol. Additional details on the Pinocchio protocol and on the other zk-SNARK protocols can be found in the original papers [7, 16, 26, 30].

3.1 Public Verifiability

We first recall the definitions of non-interactive publicly verifiable computing (see for instance [18]). Let f be a function, expressed as an arithmetic circuit over a finite field $\mathbb{F}$ and λ be a security parameter. The `Setup` procedure produces two public keys, an evaluation key EK_f and a verification key VK_f. These keys depend on the function f, but not on the inputs. The setup phase might be done once for all, and the keys are reusable for all future inputs:

$$(EK_f, VK_f) \leftarrow \texttt{Setup}(1^\lambda, f).$$

Then a prover, given some input x and the evaluation key EK_f, computes $y = f(x)$ and generates a proof of correctness π for this result:

$$(y, \pi) \leftarrow \texttt{Prove}(EK_f, x).$$

Anyone, given the input/output (x, y), the proof π and the verification key VK_f can check the proof:

$$d \in \{0, 1\} \leftarrow \texttt{Verify}(VK_f, x, y, \pi).$$

Regarding the security properties such a scheme should satisfy, honestly generated proofs should be accepted (correctness), and a cheating prover should not convince a verifier of a false computation (soundness). Formal definitions and security proofs can be found in [26].

3.2 Quadratic Arithmetic Programs

To be able to perform verifiable computation on a function f, this function has first to be expressed as an arithmetic circuit $\mathcal{F}$. Given an arithmetic circuit $\mathcal{F}$ over $\mathbb{F}_p$ with fan-in 2 gates, each multiplication gate is thus described thanks to 3 families of polynomials respectively coding the left input, the right input and the output of the gate. Addition gates, and multiplication-by-constant gates, are taken into account in these polynomials. The constraints between inputs and outputs of every gates of the circuit are captured in three families of polynomials, denoted $\mathcal{V} = (v_i(x))_i$, $\mathcal{W} = (w_i(x))_i$, $\mathcal{Y} = (y_i(x))_i$, and a target polynomial, denoted T. All these polynomials basically form a QAP. An arbitrary root $r_g \in \mathbb{F}_p$ is picked for each multiplication gate g in $\mathcal{F}$ and the target polynomial is

defined as $T(x) = \prod_{g \in \mathcal{F}} (x - r_g)$. Then, an index is assigned to each input of the circuit and to each output from a multiplication gate. During the evaluation of the circuit, the prover computes all the intermediate values of the circuit, here denoted c_i. He then computes the polynomial $P(x) = (\sum c_i v_i(x))(\sum c_i w_i(x)) - (\sum c_i y_i(x))$. If P vanishes at a root r_g picked for a multiplicative gate g, the definitions of the polynomial families implies that:

$$0 = c_{v_{r_g}} \cdot c_{w_{r_g}} - c_{y_{r_g}} \Leftrightarrow c_{v_{r_g}} \cdot c_{w_{r_g}} = c_{y_{r_g}} \tag{1}$$

(1) describes the multiplicative relation between input and output values of the gate. As a consequence, checking that T divides P is equivalent to check if P vanishes at all the roots of T and thus comes down to check all the multiplicative relations within the circuit.

3.3 The Pinocchio protocol

A full version of the protocol is given in the original paper [26]. We just sketch here its principle. Each set of polynomials $\mathcal{V}$, $\mathcal{W}$, $\mathcal{Y}$ of the QAP is mapped to an element in a bilinear group. For instance, let $V_k \in \mathcal{V}$ for some k, we have $V_k \in \mathbb{F}_p[X]$. An element of the form $g^{V_k(s)}$ is added to the public evaluation key, where g is a generator of the group and s is a secret value randomly chosen at the setup. Then, for a given input, the prover evaluates the circuit directly to obtain the output and the values of the internal circuit wires. These values are then used to build the coefficients of the QAP polynomial P, see Sect.3.2. Let c_i denote these coefficients. The prover evaluates $g^{V(s)} = \prod_k (g^{V_k(s)})^{c_k}$, and similarly for $g^{W(s)}, g^{Y(s)}$. After computing $H = P/T$, he is able to compute $g^{H(s)}$ thanks to some elements in the evaluation key. The proof sent to the verifier roughly consists of $(g^{V(s)}, g^{W(s)}, g^{Y(s)}, g^{H(s)})$. The verifier uses a bilinear pairing to check the consistency of the proof. Some additional relations and checks are added in order to ensure that the inputs have been integrated in the circuit by the prover, otherwise he could cheat.

3.4 Making a proof a zk-SNARK

In the Pinocchio protocol, the proof can be turned into a zk-SNARK with little additional computations. Let assume that the computation to verify is $y = f(x, w)$, where x is an input supplied by the verifier, f is the function on which the verifier and the prover agreed and w is a private input of the prover. Loosely speaking, in a zero-knowledge proof the prover can convince the verifier that *he knows some value* w *such that*: $y = f(x, w)$. This is done without giving information about w to the verifier, nevertheless the proof is still verifiable. Technically, this goal is achieved by adding a random multiple of the target polynomial T to every polynomial of $(\mathcal{V}, \mathcal{W}, \mathcal{Y})$, in such a way that the checks still hold if, and only if, the circuit with the given input/output is satisfied.

3.5 Expressivity of zk-SNARK schemes

Even if efficient zk-SNARKs protocols exist, there are still several challenges to be solved to reach full practicality. One of them is the expressiveness of the protocols, i.e. their capacity to verify large class of programs. The Pinocchio protocol cannot verify data dependant loops and is not efficient in branching programs because it has to evaluate both branches. Note that some protocols like TinyRAM [7] or Buffet [30] solve this issue but the representation of these computations as arithmetic circuits generates overheads.

3.6 Security

The q-power knowledge of exponent (q-PKE) and q-power Diffie-Hellman (q-PDH assumptions) have been defined by Groth [21] and the q-strong Diffie-Hellman (q-SDH) by Boneh and Boyen [10].

Parno et al. [26] show that if the QAP computed from the circuit to verify has d multiplicative gates, their protocol is sound (see Sect. 3.1) under the d-PKE, (4d+4)-PDH and (8d+8)-SDH assumptions.

4 An instantiation of the VDR scheme

We now introduce a possible instantiation of the **VDR** scheme, keeping in mind that we seek efficiency for the prover. We consider that documents are represented as gray-scale images, modeled as matrices of $n \times n$ pixels. Pixels values vary between 0 (black) and 255 (white). Redacting a part of the document thus means that pixels of the redacted area are turned black, so the symbol # defined in Sect. 2.2 is the pixel value 0. The set MOD of redacted parts of the image is therefore a set of coordinates, which locates the redacted pixels positions.

We consider implemented verifiable computation schemes to instantiate our scheme, more specifically the scheme base on Parno et al. protocol [26, 7]. The verification is efficient and the schemes based on QAPs (Sect. 3) have a short, constant-length proof that is quick to verify. The difficulty is the prover's computational overhead, which is linked to the number of multiplication gates in the arithmetic circuit representing the function to verify. More precisely, the prover's work has complexity $O(N \log^2 N)$, where N is the circuit size [26]. Therefore an efficient arithmetic circuit has first to be designed to limit the number of multiplicative gates.

4.1 The arithmetic circuit design.

To build the proof used in the `Redact` algorithm of the **VDR** scheme (Sect. 2.2), an arithmetic circuit representing the computation to verify has to be designed in order to apply the Parno et al. protocol [26] (some background on this protocol can be found in Sect. 3). A high level view of this circuit is described in Fig. 1. It contains two sub-circuits verifying respectively the value of the hash passed by

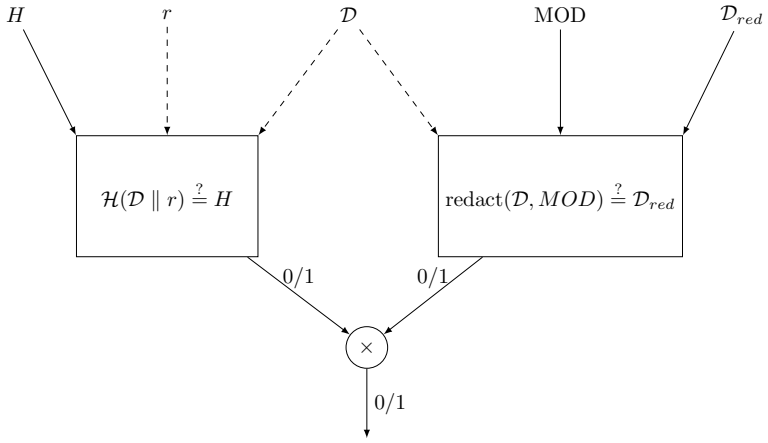

Fig. 1. Arithmetic circuit computing the proof in `Redact`. A dashed arrow means that the input is private (and supplied by the prover).

the document issuer to the client and the comparison between the original and the redacted documents. The operations involved in the sub-circuits are crucial for the prover efficiency. The circuit has to be carefully designed to be able to redact several document on different places and to amortize the key generation cost over several proof computation. Moreover, if the circuit which verify the correct redaction in the `Redact` algorithm is changed for some document, the evaluation and verification keys will change and have thus to be exchanged with the service provider. We designed a circuit able to prove the correct redaction for every document modeled as an image of size $n \times m$.

The verification of the hash signature used by the document issuer is not part of the proof for efficiency reasons. Backes et al. [5] present a verifiable computing scheme suited for working with authenticated data but, even if the performance are better than verifiying signature with the Pinocchio scheme [26], the verification of the signature is way more efficient if it is done outside the proof. Besides, since the proof requires the hash of the document there is an explicit link between the redacted document and the original one. The addition of the hash to the proof only slightly increases the length of the proof. The verifier thus first verify the signature of the hash value to test whether it indeed correspond to a value generated by the document issuer. If the verification passes, the verifier can use this hash value as input for the proof verification.

Document Redaction Since the document to redact is modeled as a matrix, the proof described in Sect. 2.2 can be represented as an arithmetic circuit using a boolean matrix for the MOD set. The function in the verifiable computing scheme for which we compute a proof takes as input a redacted document $\mathcal{D}_{red}$, a hash value H and a set MOD. We denote by $d_{i,j}$ (resp. $d^{red}_{i,j}$) the pixel in position

(i, j) of $\mathcal{D}$ (resp. $\mathcal{D}_{red}$). The prover supplies as private input the document $\mathcal{D}$ and the value r, the function f returns the value $d \in \{0, 1\}$ which is the product of the following boolean tests: $\begin{cases} \exists r, \mathcal{D} \text{ such that: } C \stackrel{?}{=} H(\mathcal{D} \parallel r) \\ \forall (i,j) \in MOD : d_{i,j}^{red} \stackrel{?}{=} 0 \\ \forall (i,j) \notin MOD : d_{i,j} - d_{i,j}^{red} \stackrel{?}{=} 0 \end{cases}$

Using a boolean matrix M as a mask, we can rewrite the two last set of tests in a more uniform way. We define the matrix $M = (m_{i,j})$ as: $m_{i,j} = 0$ if pixel (i, j) is redacted and $m_{i,j} = 1$ otherwise. The tests can thus be rewritten as: $\forall (i, j) \in \{1, \ldots, n\}^2, d_{i,j} \times m_{i,j} \stackrel{?}{=} 0$. This leads to a small arithmetic sub-circuit to check if the redacted document has not been modified in other places that the given ones.

Hash function. The proof computed by `Redact` contains the verification of the hash value computed from the original document so we need to choose a hash function efficiently verifiable i.e. a function which can be represented as an arithmetic circuit with few gates. Hash function building on the subset sum problem are well suited for arithmetic circuits [12, 8]. They were introduced by Ajtai [3] and proved collision-resistant by Goldreich et al. [19]. The collision resistance of the hash function relies on the hardness of the Short Integer Solution (SIS) problem. We first recall the Ajtai hash function.

Definition 2. *Let m, n be positive integers and q a prime number. For a randomly picked matrix $A \in \mathbb{Z}_q^{n \times m}$, the Ajtai hash $H_{n,m,q} : \{0, 1\}^m \to \mathbb{Z}_q^n$ is defined as:*

$$\forall x \in \{0, 1\}^m, \quad H_{n,m,q} = A \times x \mod q \tag{2}$$

A concrete hardness evaluation is studied by Kosba et al. in [24]. Choosing $\mathbb{F}_p$, with $p \approx 2^{254}$ to be the field where the computations of the arithmetic circuit are done leads to the following parameters for approximately 100 bit of security:

$$n = 3, m = 1524, q = p \approx 2^{254}.$$

We also used another finite field in our experiments with a lower security level of 80 bit for the associated elliptic curve. Following the method of [24], we obtained the following parameters:

$$n = 2, m = 724, q = p \approx 2^{181}.$$

Few gates are needed to implement an arithmetic circuit for this hash function: to hash m bits, $n \times m$ multiplicative gates are needed. With the parameters selected in [24], this means that 4572 gates are needed to hash 1524 bits. As a comparison, Ben-Sasson et al. designed a hand-optimized arithmetic circuit to verify the compression function of SHA-256 [6]. Their arithmetic circuit can therefore hash 512 bits and has about 27000 gates.

4.2 Experimental results

We implemented our protocol and benchmarked the verifiable computing part of the scheme since time consumption of the other parts is negligible compared to this one. Verifiable computation is implemented using the `libsnark` library [2]. The tests were run on a two different machines. The first one, denoted by machine 1 in the tables, is running at 3.6 GHz with 4 GB of RAM, with no parallelisation. The second one, denoted machine 2, is more powerful: it has 8 cores running at 2.9 GHz with 16 GB of RAM and uses parallelisation. We first implemented our scheme for images of size 128×128 and chose elliptic curves at a 128 bit and 80 bit security level [9]. The size of the proof is constant and short (less than 300 bytes) and thus the verification is fast. Table 1 summarizes the implementation results with machine 1. The column Constraints reports the number of constraints needed to check the satisfiability of the circuit implementing the proof redaction. For each security level of the proof, we implemented our scheme with the SHA256 hash and the Ajtai hash functions for comparison.

Table 1. Benchmark of verifiable computation in the **VDR** scheme (128×128 images, machine 1)

Security	Hash fct	Constraints	EK size	VK size	`KeyGen`	`Redact.Prove`	`DocVerif`
128 bit	Ajtai	19435	7.1 MB	1.3 MB	5.6 s	3.4 s	0.07 s
128 bit	SHA256	43920	13.7 MB	1.3 MB	9.2 s	4.7 s	0.07 s
80 bit	Ajtai	17834	5.4 MB	1.0 MB	5.5 s	2.4 s	0.07 s
80 bit	SHA256	43920	10.8 MB	1.0 MB	9.7 s	3.5 s	0.07 s

Table 2 reports implementation of the proving scheme using Ajtai hash function and a soundness security of 80 bit with variation on the image size. Table 3 reports the same implementation running on machine 2, with parallelisation. Note that even if the proof has constant size, the verification time increases with the image size. This is due to the time to parse the input redacted image to compute some elements to verify the proof. We continued our experiments until we reached approximately the size of an A4 document scanned at 100 dpi: we tested a 1200×800 image while the true size of an A4 document scanned at 100 dpi would be 1169×827.

The prover has most of the computational work to do with the `Redact` algorithm. However, this does not affect the practicality of the **VDR** scheme in the case of image redaction. Indeed, the proof is non-interactive and the client can prepare its redacted document, compute the related proof and submit both later to a service provider. On the service provider side, the verification is fast and does not require to share any secret with the client. The time to verify the signature of the hash value C has to be added to the verification time given in Table 1. Using a simple benchmark on OpenSSL, the time reported for signature verification is less than 1 ms for RSA signatures and ECDSA signatures with

the same computer. We conclude that the **VDR** scheme is compatible with a practical use.

Table 2. Scaling experiment of the proving part in the **VDR** scheme (machine 1 + no parallelisation)

Image size	Constraints	EK size	VK size	`KeyGen`	`Redact.Prove`	`DocVerif`
128×128	17834	5.4 MB	1.0 MB	5.6 s	2.4 s	0.07 s
400×400	161450	47.9 MB	9.9 MB	38.8 s	20.7 s	0.51 s
500×500	251450	74.0 MB	15.5 MB	58.2 s	32.8 s	0.89 s
600×600	361450	106.0 MB	22.3 MB	81.1 s	50.8 s	1.3 s
1200×800	961450	286.4 MB	59.5 MB	201.3 s	124.5 s	3.3 s

Table 3. Scaling experiment of the proving part in the **VDR** scheme (machine 2 + parallelisation)

Image size	Constraints	EK size	VK size	`KeyGen`	`Redact.Prove`	`DocVerif`
128×128	17834	5.4 MB	1.0 MB	1.9 s	0.8 s	0.07 s
400×400	161450	47.9 MB	9.9 MB	12.4 s	5.9 s	0.5 s
500×500	251450	74.0 MB	15.5 MB	19.7 s	9.9 s	0.82 s
600×600	361450	106.0 MB	22.3 MB	26.2 s	14.5 s	1.17 s
1200×800	961450	286.4 MB	59.5 MB	66.8 s	39.7 s	3.3 s

5 Conclusion

We designed a new scheme to redact an authenticated document, with the goal to hide sensitive or private data on document digitized as images. Our scheme is related to redactable signatures schemes but allow to get a much shorter signature which does not depend on the size of the redaction on the original document. Moreover, most of the existing redactable signature schemes could not be deployed in the image redaction use case we described. In contrast, the running time of our implementation shows that the protocol is practical for the different participants. We also note that every progress made in the efficiency of verifiable computation schemes would lead to performance improvement for our scheme.

Acknowledgments. The authors would like to thank Gaïd Revaud for her precious programming assistance and the anonymous reviewers of ESORICS for

their valuable feedback and comments. The authors would also like to thank Emmanuel Prouff for helpful comments that improved the quality of this manuscript. This work was partly supported by the TREDISEC project (G.A. no 644412), funded by the European Union (EU) under the Information and Communication Technologies (ICT) theme of the Horizon 2020 (H2020) research and innovation programme.

References

1. 2D-Doc. `https://ants.gouv.fr/Les-solutions/2D-Doc`, accessed: 2017-01-10
2. `Libsnark`. Available at `https://github.com/scipr-lab/libsnark`, accessed: 2017-04-19
3. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 99–108 (1996)
4. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM 45(1), 70–122 (1998)
5. Backes, M., Barbosa, M., Fiore, D., Reischuk, R.M.: ADSNARK: nearly practical and privacy-preserving proofs on authenticated data. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015. pp. 271–286 (2015)
6. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014. pp. 459–474 (2014)
7. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: Snarks for C: verifying program executions succinctly and in zero knowledge. In: Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. pp. 90–108 (2013)
8. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II. pp. 276–294 (2014)
9. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014. pp. 781–796 (2014), `https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson`
10. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. pp. 56–73 (2004)
11. Boneh, D., Shoup, V.: A graduate course in applied cryptography, version 0.3. `http://cryptobook.us` Accessed: 2017-01-15
12. Braun, B., Feldman, A.J., Ren, Z., Setty, S.T.V., Blumberg, A.J., Walfish, M.: Verifying computations with state. In: ACM SIGOPS 24th Symposium on Operating Systems Principles, SOSP'13, Farmington, PA, USA, November 3-6, 2013. pp. 341–357 (2013)

13. 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. pp. 87–104 (2010)
14. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings. pp. 310–331 (2010)
15. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 90–112 (2012)
16. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile verifiable computation. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015. pp. 253–270 (2015)
17. Derler, D., Pöhls, H.C., Samelin, K., Slamanig, D.: A general framework for redactable signatures and new constructions. In: Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers. pp. 3–19 (2015)
18. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. pp. 626–645 (2013)
19. Goldreich, O., Goldwasser, S., Halevi, S.: Collision-free hashing from lattice problems. Electronic Colloquium on Computational Complexity (ECCC) 3(42) (1996), http://eccc.hpi-web.de/eccc-reports/1996/TR96-042/index.html
20. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 113–122 (2008)
21. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. pp. 321–340 (2010)
22. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short pcps. In: 22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA. pp. 278–291 (2007)
23. 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. pp. 244–262 (2002)
24. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., abhi shelat, Shi, E.: CØcØ: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093 (2015), http://eprint.iacr.org/2015/1093
25. Naveh, A., Tromer, E.: Photoproof: Cryptographic image authentication for any set of permissible transformations. In: IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016. pp. 255–271 (2016)

26. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013. pp. 238–252 (2013)
27. Setty, S.T.V., McPherson, R., Blumberg, A.J., Walfish, M.: Making argument systems for outsourced computation practical (sometimes). In: 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012 (2012)
28. Slamanig, D., Rass, S.: Generalizations and extensions of redactable signatures with applications to electronic healthcare. In: Communications and Multimedia Security, 11th IFIP TC 6/TC 11 International Conference, CMS 2010, Linz, Austria, May 31 - June 2, 2010. Proceedings. pp. 201–213 (2010)
29. 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. pp. 285–304 (2001)
30. Wahby, R.S., Setty, S.T.V., Ren, Z., Blumberg, A.J., Walfish, M.: Efficient RAM and control flow in verifiable outsourced computation. In: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015 (2015)
31. Walfish, M., Blumberg, A.J.: Verifying computations without reexecuting them. Commun. ACM 58(2), 74–84 (Jan 2015), http://doi.acm.org/10.1145/2641562

A Appendix

We prove Theorem 1 in this section. We will prove that our **VDR** scheme is private (Lemma 1) and unforgeable (Lemma 2), which will imply Theorem 1.

Lemma 1. *If the VC scheme provides (statistical) zero-knowledge proofs and the hash function H is such that $H_r := H(.,r)$ is a secure PRF then the **VDR** scheme is private.*

Proof. We will bound the advantage of a PPT adversary attacking the privacy of the scheme using a sequence of games. More precisely we will show that $\mathtt{Adv}^{A}_{\mathrm{Leak}}$ is negligible.

Game 0 This is the original `Leak` game.

Game 1 Same as Game 0 but here the oracle $\mathcal{O}^{\mathtt{Auth/Redact}}$ picks a random value h, signs it and returns the couple h, σ, instead of C_b, σ. Let S_1 be the event that $b^\star = b$ in Game 1. Since $H(.,r)$ is assumed to be a secure PRF, we have that: $Pr\left[S_0\right] - Pr\left[S_1\right] \leqslant \epsilon_{PRF}$, where ϵ_{PRF} is the PRF advantage.

Game 2 Same as Game 1, but the part of the oracle $\mathcal{O}^{\mathtt{Auth/Redact}}$ computing the proof is replaced by the simulator. Let S_2 be the event that $b^\star = b$ in Game 2. We have that $Pr\left[S_2\right] = Pr\left[S_1\right]$

Game 3 Same as Game 2, but the simulator of the oracle $\mathcal{O}^{\mathtt{Auth/Redact}}$ outputs its proof π without having knowledge of the messages $M_c, c \in \{0,1\}$. Let S_3 be the event that $b^\star = b$ in Game 3. Since the VC scheme is assumed to be zero-knowledge, we have that there exists a negligible function ϵ_{SZK} such

that: $Pr\left[S_3\right] - Pr\left[S_2\right] \leqslant \epsilon_{SZK}$. Since the signature is now only composed of random elements, we have $Pr\left[S_3\right] = \frac{1}{2}$.
Gathering the results of all the games, we finally conclude that:

$$|Pr\left[S_0\right] - \frac{1}{2}| \leqslant \epsilon_{PRF} + \epsilon_{SZK} \tag{3}$$

Therefore the **VDR** scheme is secure. □

Lemma 2. *If the VC scheme is sound and the signature scheme is EUF-CMA, then the **VDR** scheme is unforgeable.*

Proof (sketch). We show if there exists an efficient adversary succeeding in the `Forge` experiment, denoted by $\mathcal{A}_{\texttt{Forge}}$, we can build an efficient adversary $\mathcal{A}_{\texttt{EUF-CMA}}$ breaking the EUF-CMA property of the signature or an efficient adversary $\mathcal{A}_{\texttt{VC}}$ breaking the soundness of the verifiable computing scheme. These adversaries are built by forwarding the queries made by $\mathcal{A}_{\texttt{Forge}}$. At the end, $\mathcal{A}_{\texttt{Forge}}$ outputs a redacted forged document, which is not part of the queries made before. This redacted forged document is also a forgery for the signature scheme or for the verifiable computation scheme.