Final Version for Eurocrypt '96, Zaragoza, May 1996, proceedings to appear in LNCS, Springer-Verlag

# Asymmetric Fingerprinting

(Extended Abstract)

Birgit Pfitzmann*, Matthias Schunter†

Universität Hildesheim, Institut für Informatik
D-31141 Hildesheim, Germany

**Abstract**. Fingerprinting schemes deter people from illegal copying of digital data by enabling the merchant of the data to identify the original buyer of a copy that was redistributed illegally. All known fingerprinting schemes are symmetric in the following sense: Both the buyer and the merchant know the fingerprinted copy. Thus, when the merchant finds this copy somewhere, there is no proof that it was the buyer who put it there, and not the merchant.

We introduce asymmetric fingerprinting, where only the buyer knows the fingerprinted copy, and the merchant, upon finding it somewhere, can find out and prove to third parties whose copy it was. We present a detailed definition of this concept and constructions. The first construction is based on a quite general symmetric fingerprinting scheme and general cryptographic primitives; it is provably secure if all these underlying schemes are. We also present more specific and more efficient constructions.

## 1 Introduction

Where information is sold, the merchants want to ensure that it is not redistributed illegally on a large scale. Thus copyright protection of digital data has recently found increased attention given the widespread feeling that electronic marketplaces, mainly on the World Wide Web, may soon be reality. In particular, images and text are now considered in addition to software, where the problem has been relevant much longer. There are two basic approaches to copyright protection. One is based on tamper-resistant modules in the buyers' machines that prevent copying at least of the internal representation of the data. However, this approach is limited, as experiences with software have shown. The second approach, fingerprinting, does not rely on special hardware. Thus it cannot prevent copying, but it deters people from illegal copying by allowing a redistributed copy to be traced back to its original owner.

### 1.1 Fingerprinting

Fingerprinting software, images, or other data means introducing some errors into each copy sold that make this copy unique, as fingerprints make people unique. These intentional errors are called *marks*, and all the marks in one copy are *the fingerprint*. Once an illegal copy turns up, the merchant can see from the fingerprint which of the original copies was illegally redistributed. Usually, the following informal requirements are posed on such a scheme:

* pfitzb@informatik.uni-hildesheim.de

† schunter@acm.org

- The data must tolerate the errors: On the one hand, the marks must not decrease the usefulness of the copy to the buyer. On the other hand, the buyer should not be able to derive from redundancy in the data where the marks are.
- Collusion-tolerance: Even if dishonest buyers have up to a certain number of copies, they should not be able to find all the marks by comparing the copies. In particular, the fingerprints must have a common intersection.
- Tolerance of additional errors: If a dishonest buyer adds some noise to the copy, the fingerprint should still be recognizable, unless there is so much noise that the copy as such is useless. In other words, the fingerprints should tolerate a greater level of noise than the data. In particular, this should hold for lossy data compression.

For previous literature on fingerprinting, see, e.g., [W83, BMP86, C95, BS95]. Related problems are treated in [CFN94] for Pay-TV, in [BLM94, CMP95] for data that are not digital, but on paper, and in [MQ95, Section V], where a so-called watermark that proves the identity of the merchant is desired. Examples of cryptographic work on software copyright protection with special hardware are [G87, O90].

### 1.2 Asymmetric Fingerprinting

In all the previous fingerprinting schemes, two parties know the data with the fingerprint: the buyer of the original copy and the merchant. Thus, if another copy with this fingerprint turns up, one cannot really assign responsibility to one of them: The copy might just as well have been redistributed by a dishonest employee of the merchant as by the buyer, or the merchant as such may want to gain money by wrongly claiming that there are illegal copies around. In other words, the merchant does not obtain means to prove to a third party that the buyer redistributed the copy.

This is as with symmetric message authentication codes: As two people know the same secret key, no non-repudiation is offered. In contrast, only one person can make a digital signature (at least if we neglect problems with hardware protection and key distribution), and thus a signed message can really be used to hold someone responsible. We want to achieve similar security for fingerprinting.

Thus our goal is to construct fingerprinting schemes that are asymmetric in the following sense: After a sale, *only* the buyer knows the data with the fingerprint. However, if the merchant later finds this copy somewhere, he can identify the buyer and prove to third parties that this buyer bought this copy.

## 2 The Model

To make our goals and assumptions precise, we give a formal definition of asymmetric fingerprinting. We first define the components of such a scheme and then the security requirements.

### 2.1 Algorithms and Parameters

A fingerprinting scheme works for a specific space of data to be sold. We call it *Pic_Space*, because in the examples and illustrations, we mostly consider pictures. As the original data the merchant sells have to remain secret, and the buyers may have some a priori knowledge about it, we have to assume that *Pic_Space* is equipped with an arbitrary probability distribution. One buyer, identified by a public key that is constant over some time, may buy several data from the same merchant. We use an input parameter *text* to the fingerprinting protocol to distinguish these sales. In practice, *text* would typically identify the contract referring to this sale. This is particularly important if different data are sold with different licensing terms.

**Definition 1.** An asymmetric fingerprinting scheme consists of four protocols, *key_gen*, *fing*, *identify*, and *dispute*. All algorithms in these protocols have a security parameter, $k$, as a common input, and are probabilistic polynomial-time in $k$. Furthermore, the scheme contains *Pic_Space*, the space of data to be sold, with a given probability distribution, and *Text_Space*; both are subsets of $\{0, 1\}^*$. For simplicity, we have omitted in the notation that *Pic_Space* and *Text_Space* may depend on $k$. Moreover, there may be a parameter $N$ denoting the maximum number of times the merchant can sell the same data. The protocols have the following parameters:

- For *key_gen*, the key generation protocol, we only define the simplest case: The buyer generates a pair of values $(sk_B, pk_B)$, which we call a secret and a public key, and broadcasts $pk_B$ reliably to all merchants and third parties, e.g., via certification authorities.
- *fing*, the fingerprinting protocol, is a 2-party protocol between a merchant, $M$, and a buyer, $B$; see Figure 1. The merchant inputs the data to be sold, *Pic*, the identity of the buyer, represented by the buyer's public key, $pk_B$, and a string $text \in Text\_Space$. Moreover, his algorithm may depend on the history, represented by $Record\_list_{Pic}$, a list of records from previous sales of the same data. The buyer inputs *text* and her secret key, $sk_B$. The main output to the buyer are the data $Pic_{bought}$ with a small error. The buyer may also obtain a record, $record_B$, to be stored for future disputes. Instead, she may obtain a specific output *failed*, which means that the protocol failed. The output for the merchant is a record of the sale, $record_M$, or *failed*.

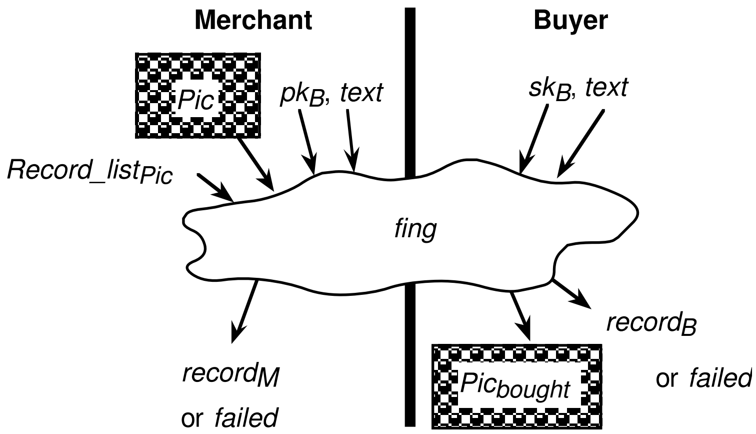


**Fig. 1.** Parameters of asymmetric fingerprinting

- *identify* is an algorithm the merchant executes to identify the original buyer of a certain copy, see Figure 2. The inputs are the data with a small error, $Pic_{found}$, the data the merchant sold, *Pic*, and a list $Record\_list_{Pic}$ of all the records of sales of these data. The output is either *failed* or the identity of a buyer, represented by $pk_B$ and *text*, and a string *proof*.

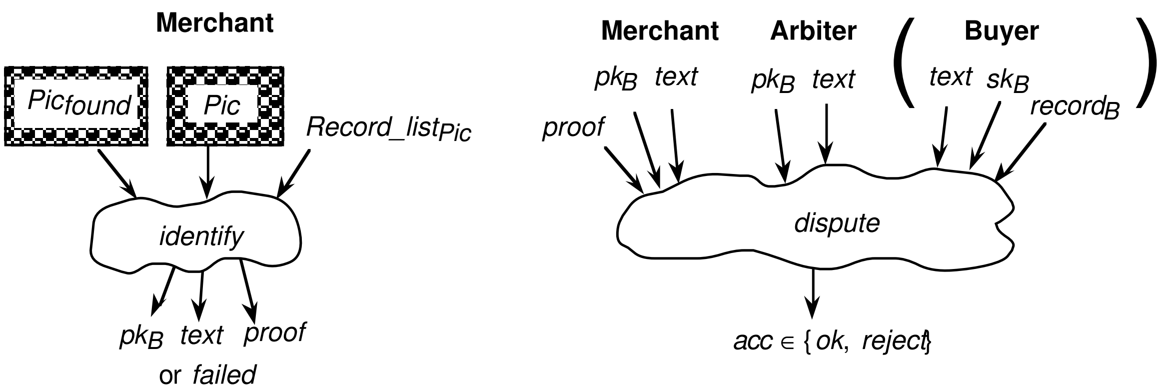


**Fig. 2.** Parameters in the identification of the original owner and the subsequent dispute

- *dispute* is a 2- or 3-party protocol between the merchant, an arbitrary third party that we briefly call *arbiter*, and possibly the accused buyer. The merchant and the arbiter input $pk_B$ and *text*. The merchant additionally inputs *proof*. If the accused buyer takes part, she inputs *text*, her secret key, $sk_B$, and $record_B$, the record from the purchase with *text*. The output is a Boolean value *acc* for the arbiter, where $acc = ok$ means that the arbiter finds the buyer guilty or, more precisely, that the arbiter accepts that the merchant has found the data bought by the accused buyer with *text*. ♦

## 2.2 Security

There are three types of requirements on an asymmetric fingerprinting scheme: First, it should be effective in the sense that buyers obtain useful data as long as nobody cheats. Secondly, merchants want to be protected from cheating buyers: If an illegal copy turns up, they want some buyer to be identified and found guilty as responsible for it. Thirdly, buyers want to be protected from cheating merchants and other buyers: If they do not redistribute their copies illegally, they should not be falsely found guilty.

The notion of "useful data" that the buyer is supposed to obtain depends on the type of data to be fingerprinted. Thus we assume an arbitrary given relation $similar_1$, where $similar_1(Pic, Pic_{bought})$ means that $Pic_{bought}$ is good enough as a replacement for *Pic*. For instance, with pictures, $similar_1$ will typically mean $Pic_{bought} = Pic \oplus \varepsilon_{bought}$ with an error vector of small Hamming weight. With programs, it may mean semantic equivalence. Similarly, we assume an arbitrary given relation $similar_2$ where $similar_2(Pic, Pic_{found})$ means that an illegal copy $Pic_{found}$ is still so close to *Pic* that the merchant wants to identify the original buyer.

The protection of the merchant usually only works for collusions of a certain maximum size, just as with symmetric fingerprinting. In contrast, we guarantee the protection of a buyer even if any number of others collude against her, because falsely being found guilty of fraud would be a completely unacceptable consequence of the use of fingerprinting.

Now we are ready for the formal definitions.

**Definition 2.** A fingerprinting scheme is called **effective** for a certain given binary relation $similar_1$ on *Pic_Space* iff the following holds: If both parties execute *fing* honestly for any $Pic \in Pic\_Space$ and $text \in Text\_Space$, after correct key generation and any intermediate history, the protocol succeeds and the buyer obtains a close enough variant $Pic_{bought}$ of *Pic*, i.e., $similar_1(Pic, Pic_{bought})$ holds. ♦

**Definition 3.** A fingerprinting scheme is called **secure for the merchant under** ***coll_size*** **collusions** and for a given binary relation $similar_2$, where $coll\_size$ is a function: $\mathbb{N} \rightarrow \mathbb{N}$, iff the following holds for all probabilistic polynomial-time interactive algorithms $\tilde{B}$:

- If the merchant selects data *Pic* from *Pic_Space* and executes *fing* for these data with $\tilde{B}$ at most *coll_size*($k$) times, where $\tilde{B}$ can choose the merchant's input *text* arbitrarily and $pk_B$ among a set of keys that the merchant has accepted as distributed, and where key generation and disputes with $\tilde{B}$ and interactions with other buyers may occur in between,
- and if $\tilde{B}$ then outputs data $Pic_{found}$ similar to *Pic*, i.e., such that $similar_2$(*Pic*, $Pic_{found}$) holds,
- then *identify*, on input $Pic_{found}$, *Pic*, and the current list of records, outputs a triple ($pk_B$, *text*, *proof*), where $pk_B$ is one of the previous inputs to *fing*, such that
- when *dispute* is executed with these parameters (input by the merchant and the arbiter), and in the 3-party-case with $\tilde{B}$, the arbiter's output will be *ok*, except with negligible probability.

The probability is over all the random choices of all the participants. ♦

**Definition 4.** A fingerprinting scheme is called **secure for the buyer** iff the following holds: No probabilistic polynomial-time algorithm $\tilde{M}$

- carrying out *fing* with *B* arbitrarily often (after *B* has generated a key pair and distributed $pk_B$), where $\tilde{M}$ can choose the buyer's input *text* arbitrarily for the same or different data, and possibly carrying out some disputes in between,
- and obtaining the outputs $Pic_{bought}$ from some of these executions, which $\tilde{M}$ may choose adaptively by inputting the corresponding *text*,
- can compute parameters *text* and *proof*, where *text* is not among those for which $\tilde{M}$ obtained $Pic_{bought}$ in the previous step,
- and then execute *dispute* with an arbiter that has the inputs $pk_B$ and *text* such that the arbiter's output is *ok*, except with negligible probability.

The probability is over all the random choices of all the participants. ♦

We could distinguish whether parallel or only sequential executions of *fing* and other protocols are allowed in both definitions. Furthermore, we can make several additional requirements that are useful at least in some applications, e.g., protecting merchants from making wrong accusations and making disputable whether a merchant delivered the promised information.

## 3 A General Construction

Our first construction relies on a quite general given symmetric fingerprinting scheme and some further general cryptographic primitives and is provably secure if all these primitives are secure. The buyer need not participate in disputes, nor store records. More efficient, less general constructions are shown in later sections. By an arbitrary symmetric fingerprinting scheme, we mean one according to the following very general definition:

**Definition 5 (Sketch).** A symmetric fingerprinting scheme consists of two algorithms, *fing* and *identify*, a data space, *Pic_Space*, with a given probability distribution, and an identity space, *Id_Space*. Both algorithms are assumed to be executed by the merchant. Their parameters are shown in Figure 3.

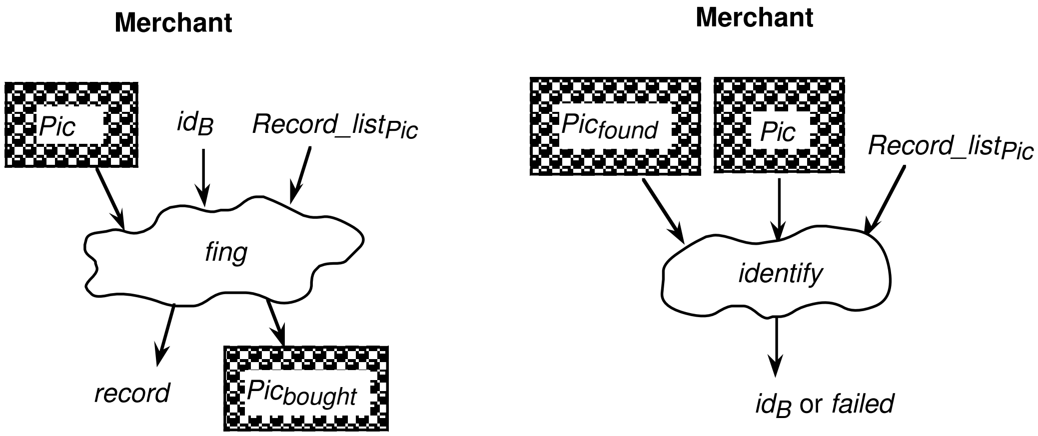


**Fig. 3.** Symmetric fingerprinting, general model

We require that such a scheme is **effective** for a relation $similar_1$ and **secure under *coll_size* collusions** for a relation $similar_2$; these definitions are similar to Definition 2 and 3. ♦

**Definition 6.** The **restrictions** we have to make on the symmetric fingerprinting scheme for Construction 1 are:

a) Memory-less. The algorithm *fing* does not produce records depending on $id_B$. Thus the whole content of $Record\_list_{Pic}$ can be derived from *Pic* and the merchant's string of random bits. For explicitness, we denote it by $Record_{Pic}$. In practice, $Record_{Pic}$ will typically be the secret tuple of mark positions used with *Pic.*

b) Sufficiently large identity space. *Id_Space* is simply $\{0, 1\}^{id\_len(k)}$ for some function *id_len* of the security parameter *k*, and large enough to serve as the preimage of a one-way function. In theory, $id\_len(k) \geq k^{\varepsilon}$ for some $\varepsilon > 0$ suffices. In practice, symmetric fingerprinting schemes typically do not have a parameter *k* for cryptographic security; their parameters are *id_len* and *coll_size*, and then the minimum size *L* of elements of *Pic_Space* is derived. Then we need that *L* is polynomial in *id_len*. This is the case, e.g., in the explicit examples in [C95] and the main scheme in [BS95]. ♦

In the following construction, we use a general 2-party protocol for secure evaluation of an arbitrary function; see, e.g., [Y86, CDG88, GMW87]. We do not particularly require either party to be unconditionally secure, nor do we require gradual release of a secret.

**Construction 1.** Let a symmetric fingerprinting scheme with the above restrictions, a general 2-party protocol, a signature scheme with algorithms *sign* and *verify*, and a one-way function *f* be given. In the notation, we distinguish the given symmetric scheme and the asymmetric scheme to be constructed by indices *sym* or *asym*, respectively.

$\boldsymbol{key\_gen_{asym}}$ is the key generation of the given signature scheme.

$\boldsymbol{fing_{asym}}$. For the first buyer, the merchant selects $Record_{Pic,sym}$ based on *Pic* and stores it as a part of $Record\_list_{Pic,asym}$. Any buyer first chooses an identity $id_{sym} \in Id\_Space_{sym}$

randomly. Then the buyer and the merchant execute a 2-party protocol for secure evaluation of the function shown in Figure 4. The merchant obtains his output first.

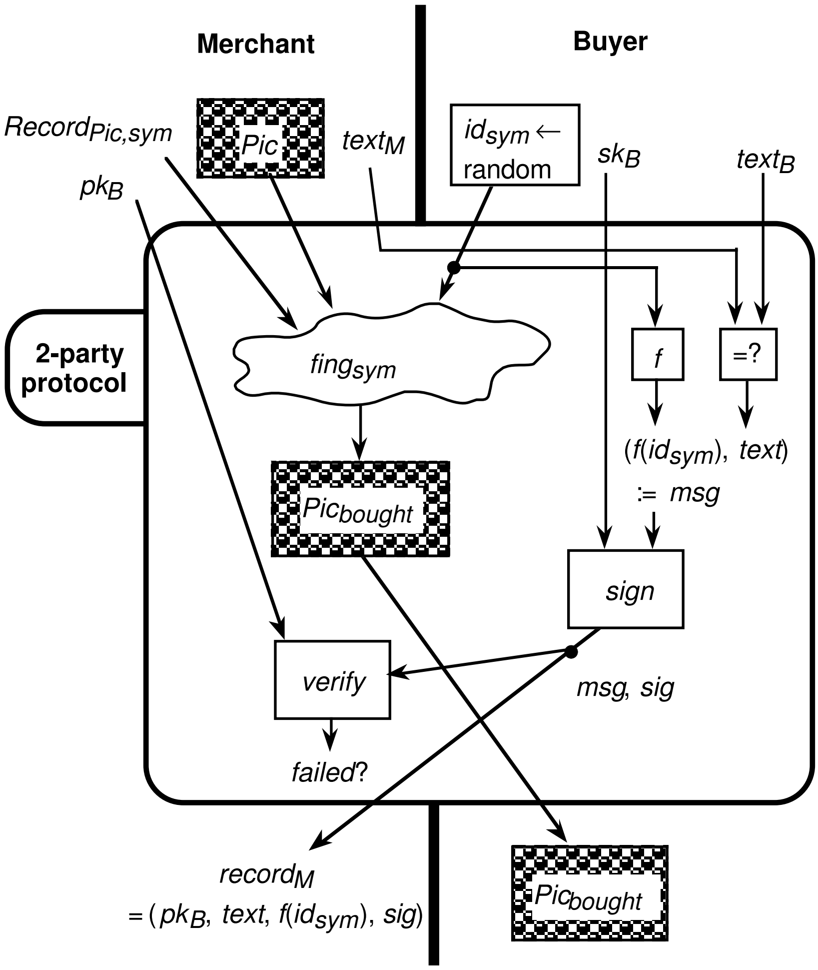


**Fig. 4.** General construction of asymmetric fingerprinting from a symmetric scheme

***identify***$_{asym}$, on input $Pic_{found}$, $Pic$, and $Record\_list_{Pic,asym}$, works as follows:

1. It executes $identify_{sym}(Pic_{found}, Pic, Record_{Pic,sym})$. If the output is *failed*, it stops. Otherwise, one obtains a value $id^*_{sym}$.
2. It computes $f(id^*_{sym})$ and searches for a record with $f(id_{sym}) = f(id^*_{sym})$.
3. If such a record is found, the output consists of the values $pk_B$ and *text* from this record and $proof := (id^*_{sym}, sig)$.

***dispute*.** The arbiter, which has the input $(pk_B, text)$ and obtains a value *proof* from the merchant, verifies that *proof* is of the form $(id_{sym}, sig)$ and that *sig* is a valid signature with respect to $pk_B$ on $msg = (f(id_{sym}), text)$. ♦

**Theorem 1.** Assume that all the underlying primitives in Construction 1 are secure, and in particular, that the 2-party protocol is secure in the sense that it can be used in the place of an oracle computing the same function. Then Construction 1 is a secure asymmetric fingerprinting scheme with the same function *coll_size* and the same relations $similar_1$, $similar_2$ as the given symmetric scheme. ♦

All the proof sketches had to be omitted in this extended abstract.

Note that with some unsolved problems about definitions of secure computation, see [B91, MR91], our requirement on the 2-party protocol, or at least on its proof, is quite strong, even though we do not require simultaneous outputs; to our knowledge no such proof has been fully carried out. Nevertheless, it is what such protocols are intuitively intended for and thus supposed to achieve.

# 4 More Specific Construction

We now derive more efficient asymmetric fingerprinting schemes from a more specific class of symmetric fingerprinting schemes. This class can be seen as typical for black-and-white pictures, except that we still make the restrictions from Definition 6.

**Definition 7.** The **restricted symmetric fingerprinting for black-and-white pictures** is based on two algorithms, *place_marks* and *code*:

- *place_marks* is a probabilistic algorithm that, on input a picture *Pic*, selects a tuple *marks* = ($mark_1$, …, $mark_l$) of suitable positions for marks. Each mark is a subset of all the pixel positions of *Pic*, say $\{1, \ldots, L\}$, and will be used to encode one bit. Any pair of two marks is disjoint.
- *code*: $Id\_Space \rightarrow \{0, 1\}^l$ is a deterministic algorithm. Its purpose is to encode the original identities such that certain collusions cannot simply find all the marks by comparing their versions of a picture.

$fing_{sym}$ is constructed from these components as follows: For the first buyer, the merchant selects *marks* with *place_marks*(*Pic*) and stores it once and for all as $Record_{Pic}$. The picture $Pic_{bought}$ for an individual buyer, $id_B$, is $Pic \oplus \varepsilon$ with

$$\varepsilon(pixel) = 1 \;\Leftrightarrow\; pixel \in mark_i \text{ for some } i \text{ with } code(id_B)(i) = 1.$$

$identify_{sym}$ has the inputs $Pic_{found}$, *Pic*, and *marks*.

*Id_Space* must be $\{0, 1\}^{id\_len(k)}$ with sufficiently large *id_len*, where in theory, $id\_len(k) \geq k^{\varepsilon}$ is again sufficient, and the use in practice can be seen in Construction 2 below. ♦

For instance, in [C95], the marks are essentially small squares, chosen according to characteristics of *Pic*, and encoding a “1” means making the square slightly darker. For black-and-white pictures, this means changing certain pixels in the square from “0” to “1”. The set of these pixels within one square, which is fixed given *Pic*, is a mark.

The following construction relies on a homomorphic bit commitment scheme, see [BCC88]. For brevity, we restrict ourselves to the quadratic residuosity scheme: One party chooses a Blum integer $n = pq$, i.e., $p$ and $q$ are primes with $p \equiv q \equiv 3 \bmod 4$. Commitments to 0 are quadratic residues, commitments to 1 are quadratic non-residues with Jacobi symbol +1. The party who has chosen $n$ can decode all commitments. Even the other party can encode 0 as $x^2$ and 1 as $-x^2$ with random $x$. $BC(s)$ for a string $s$ means the concatenation of commitments to the bits of $s$.

Furthermore, as we still have an abstract mapping *code* in the symmetric scheme, we need an abstract zero-knowledge proof system, but for a much smaller problem than the 2-party protocol was needed in Construction 1. An example instantiation is shown below.

**Construction 2 (BC Fingerprinting).** Let a symmetric fingerprinting scheme according to Definition 7, the quadratic residue bit commitment scheme, and a zero-knowledge proof system be given. Then an asymmetric fingerprinting scheme is defined as follows: We split *id_len*($k$) into a field of length $len_1 = \lceil \log_2(N) \rceil$ and the rest of length $len_2$, which is still at least of some order $k^{\varepsilon}$.

***key_gen****asym* is the key generation of the given signature scheme.

***fing****asym*. For the first buyer, the merchant executes *place_marks*(*Pic*) and stores the result *marks* as a part of *Record_list*$_{Pic,asym}$. Moreover, he initializes a $len_1$-bit counter *seq_no* with zero. Then, for any buyer:

1. The merchant sends the current value of *seq_no* to the buyer and increments *seq_no*.
2. The buyer chooses an instance $n_B$ of the bit commitment scheme. She gives the merchant a zero-knowledge proof that $n_B$ is at least a generalized Blum integer, i.e., of the form $p^r q^s$ with $p$, $q$ as above and $r$, $s$ odd [GP88].
3. The buyer chooses a value *id_proof* $\in \{0, 1\}^{len_2}$ randomly and sets $id_{sym}$ := (*seq_no*, *id_proof*) $\in$ *Id_Space*$_{sym}$. She encodes $id_{sym}$ according to the symmetric fingerprinting scheme and commits to the result, i.e., she computes *com* := *BC*(*code*($id_{sym}$)). She sends *msg* := (*com*, *text*) and a signature *sig* on *msg* to the merchant.
4. The buyer proves in zero-knowledge that she knows a value $id_{sym}$ whose first part is *seq_no*, and such that *com* is a commitment on *code*($id_{sym}$).
5. The merchant verifies *sig*. Then he encodes the whole picture as *BC*(*Pic*) and multiplies the obtained commitments into the picture at the mark positions. Thus, for $i = 1, \ldots, l$, he multiplies all the commitments to pixels in $mark_i$ by $com_i$ = *BC*(*code*($id_{sym}$)($i$)). He sends the result, which will be *BC*($Pic_{bought}$), to the buyer.
6. The buyer decrypts $Pic_{bought}$.

The merchant stores $record_M$ = (*seq_no*, $pk_B$, *text*, *msg*, *sig*). The buyer's $record_B$ consists of *text*, *seq_no*, $id_{sym}$, and the strings needed to open the commitments in *com*.

***identify****asym*. On input $Pic_{found}$, *Pic*, and *Record_list*$_{Pic,asym}$, first execute $identify_{sym}$($Pic_{found}$, *Pic*, *marks*). If this is successful, verify that the output is a pair $id_{sym}$ = (*seq_no*, *id_proof*) where *seq_no* occurs in some record $record_M$. If yes, retrieve $record_M$ = (*seq_no*, $pk_B$, *text*, *msg*, *sig*) and output $pk_B$, *text*, and *proof* = ($id_{sym}$, *msg*, *sig*).

***dispute****asym*

1. On input ($pk_B$, *text*), and upon receiving *proof* = ($id_{sym}$, *msg*, *sig*) from the merchant, the arbiter verifies that *sig* is a valid signature on *msg* with respect to $pk_B$ and that the second component of *msg* is *text*. If not, it stops with the output *reject*. Otherwise, it retrieves *com* from *msg*.
2. The arbiter now has to verify that *code*($id_{sym}$) is the content of the commitments *com*. As it cannot do this alone, it asks the buyer to prove her innocence. She gives the arbiter a zero-knowledge proof that the content of at least one commitment in *com* is not the corresponding bit of *code*($id_{sym}$). If the buyer succeeds in this, the arbiter outputs *reject*, otherwise *ok*. ♦

**Theorem 2.** If all the underlying primitives are secure, Construction 2 is a secure asymmetric fingerprinting scheme with the same function *coll_size* and the same relations $similar_1$, $similar_2$ as the given symmetric scheme. ♦

For specific symmetric fingerprinting schemes, we need efficient zero-knowledge proof systems for Step 4 of $fing_{asym}$ in Construction 2. We show this for a small example from [BS95].

**Example.** Consider $code = \Gamma_0$ from [BS95, Section 4]: For $m$ identities, $code(id)$ is the $id$-th row of the following table, where the length of each block is a certain parameter $d$ related to our $k$.

$$\begin{array}{cccc} 11\ldots1 & 11\ldots1 & \cdots & 11\ldots1 \\ 00\ldots0 & 11\ldots1 & & \vdots \\ \vdots & & \ddots & \\ & & & 11\ldots1 \\ 00\ldots0 & \cdots & & 00\ldots0 \end{array}$$

To gain efficiency, we do not let the buyer commit to a complete codeword. First, she only needs to commit to one bit per block, and the merchant can replicate it $d$ times. Secondly, the first $len_1$ bits of $id_{sym}$ have to be a specific value *seq_no*. Thus the buyer's codeword is one of a contiguous set of rows. All these rows start with a certain number of zeros and end with a certain number of ones. Only certain $x = 2^{len_2}$ values in the middle are not fixed and have to be hidden in commitments. Thus the buyer commits to some word $w = w_0 w_1 \ldots w_{x-1}$. She must prove in zero-knowledge that she knows $w$, and that it consists of a string of none or more zeros, followed by a string of none or more ones. The following steps are repeated $k$ times:

- The buyer makes $x$+1 random commitments to 0, followed by $x$ commitments to 1, and rotates them randomly, say, by $r \in_R \{0, \ldots, 2x\}$ steps. She sends the result, *circle*, to the merchant.
- The merchant has two choices:
  - The buyer must either open *circle* to show that it was constructed correctly.
  - Or she must prove that she can map $w$ into *circle*. This means that she sends the index $r^* \in \{0, \ldots, 2x\}$ where $w$ starts, and shows that $w_i$ equals the content of $circle_{(r^*+i) \bmod (2x+1)}$ for $i = 0, \ldots, x-1$. Equality is shown by opening the product of the two commitments to reveal zero. Furthermore she proves that $w$ does not consist of ones followed by zeros by opening $circle_{(r^*-1) \bmod (2x+1)}$, the commitment immediately before $w$ in the circle, to reveal zero. ♦

## 5 Using Higher Residues

We can user higher residues to improve the efficiency of Construction 2. Let $r$ be a fairly small prime, say, $k^*$+1 bits long, where $k^*$ might be 10, and $n = pq$ such that $r$ divides $\varphi(n)$ and $r^2$ does not. We can commit to a value from $\{0, 1, \ldots, r-1\}$ using the $r$ residue classes modulo the subgroup of $r$-th powers within $\mathbb{Z}_n^*$, if we believe that distinguishing these classes is hard [CF85]. These commitments are homomorphic with respect to addition modulo $r$. We denote them by *HRC*. More details, in particular about efficient en- and decoding, can be found in [B87].

We want to improve efficiency by a factor of $k^*$ by using these commitments on $k^*$-bit blocks of *Pic*. This may seem impossible at first sight, because we have to execute bitwise exor on the hidden values, not addition modulo $r$. However, let $HRC(Pic^*)$ be the commitment to a block of *Pic*, and let $Pic_j^*$ be the pixel that serves as the coefficient of $2^j$ when $Pic^*$ is interpreted as a binary number. Assume that this pixel lies in $mark_i$, and let $b$ be the bit of $code(id_{sym})$ encoded in this mark. The buyer sends $HRC(b)$. The merchant shifts the hidden bit to the correct position by computing $HRC(b)^{2^j} = HRC(b\,2^j)$. Now we exploit that the merchant knows $Pic_j^*$: If $Pic_j^* = 0$, he computes $HRC(Pic^*)\bullet HRC(b\,2^j) = HRC((Pic^* + b\,2^j) \bmod r)$, and if $Pic_j^* = 1$, he computes $HRC(Pic^*)/HRC(b\,2^j) = HRC((Pic^* - b\,2^j) \bmod r)$. No carry occurs in this addition or subtraction, and in particular no modular reduction, and thus these operations are the desired exors.

With a similar trick, higher residues can be used to encode operations on grayscale pictures efficiently.

## 6 Conclusion

We have introduced the notion of asymmetric fingerprinting, which gives copyright protection by identification after the fact the kind of security that asymmetric cryptography can offer. In particular, it offers non-repudiation, i.e., there is proof that one particular person was responsible for an action.

We presented constructions that show that this notion can be realized. The most efficient ones of these are not bad in the context of secure computation, i.e., there are no high-degree polynomials if the underlying code is simple enough, but obviously the message expansion is still prohibitive for any large-scale use, because data to be fingerprinted is typically large, and even symmetric fingerprinting only works well on large data.

However, we think that the problem has enough structure to give reason to hope that constructions without significant message expansions can be found in the future, in particular, by extending the ideas of Section 5.

## Acknowledgments

We thank Andreas Pfitzmann and Michael Waidner for helpful discussions.

## References

[B87] Josh Cohen Benaloh: *Verifiable Secret-Ballot Elections*; Dissertation, Yale University, September 1987.

[B91] Donald Beaver: *Secure Multiparty Protocols and Zero Knowledge Proof Systems Tolerating a Faulty Minority*; Journal of Cryptology 4/2 (1991) 75-122.

[BCC88] Gilles Brassard, David Chaum, Claude Crépeau: *Minimum Disclosure Proofs of Knowledge*; Journal of Computer and System Sciences 37 (1988) 156-189.

[BLM94] J. Brassil, S. Low, N. Maxemchuk, L. O'Gorman: *Electronic Marking and Identification Techniques to Discourage Document Copying*; Proceedings IEEE INFOCOM 94, Toronto, June 12-16, 1994, 1278-1287.

[BMP86] G. R. Blakley, C. Meadows, G. B. Purdy: *Fingerprinting Long Forgiving Messages*; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 180-189.

[BS95] Dan Boneh, James Shaw: *Collusion-Secure Fingerprinting for Digital Data*; Crypto '95, LNCS 963, Springer-Verlag, Berlin 1995, 452-465.

[C95] Germano Caronni: *Assuring Ownership Rights for Digital Images*; Proceedings VIS '95; Vieweg, Wiesbaden 1995, 251-263.

[CDG88] David Chaum, Ivan B. Damgård, Jeroen van de Graaf: *Multiparty computations ensuring privacy of each party's input and correctness of the result*; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

[CF85] Josh D. Cohen, Michael J. Fischer: *A robust and verifiable cryptographically secure election scheme*; 26th FOCS, 1985, 372-382.

[CFN94] Benny Chor, Amos Fiat, Moni Naor: *Tracing traitors*; Crypto '94, LNCS 839, Springer-Verlag, Berlin 1994, 257-270.

[CMP95] Abhijit K. Choudhury, Nicholas F. Maxemchuk, Sanjoy Paul, Henning G. Schulzrinne: *Copyright Protection for Electronic Publishing Over Computer Networks*; IEEE Network 9/3 (1995) 12-20.

[G87] Oded Goldreich: *Towards a Theory of Software Protection and Simulation by Oblivious RAMs*; 19th STOC, 1987, 182-194.

[GMW87] Oded Goldreich, Silvio Micali, Avi Wigderson: *How to play any mental game – or – a completeness theorem for protocols with honest majority*; 19th STOC, 1987, 218-229.

[GP88] Jeroen van de Graaf, René Peralta: *A simple and secure way to show the validity of your public key*; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 128-134.

[MQ95] Benoit M. Macq, Jean-Jacques Quisquater: *Cryptology for Digital TV Broadcasting*; Proceedings of the IEEE 83/6 (1995) 944-957.

[MR91] Silvio Micali, Phillip Rogaway: *Secure Computation (Chapters 1-3)*; Laboratory for Computer Science, MIT; distributed at Crypto '91.

[O90] Rafail Ostrovsky: *An efficient software protection scheme*; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 610-611.

[W83] Neal R. Wagner: *Fingerprinting*; IEEE Symposium on Security and Privacy, Oakland, 1983, 18-22.

[Y86] Andrew Chi-Chih Yao: *How to Generate and Exchange Secrets*; 27th FOCS, 1986, 162-167.