Towards Provably Robust Watermarking*

David Baelde[1,2], Pierre Courtieu[3], David Gross-Amblard[4], and
Christine Paulin-Mohring[1]

[1] LRI-PROVAL – Université Paris Sud/CNRS/INRIA, Saclay, France
[2] IT University of Copenhagen, Denmark
[3] Cédric – CNAM, Paris, France
[4] IRISA – Université Rennes 1, Rennes, France

Abstract. Watermarking techniques are used to help identify copies of publicly released information. They consist in applying a slight and secret modification to the data before its release, in a way that should be *robust*, *i.*e., remain recognizable even in (reasonably) modified copies of the data. In this paper, we present new results about the robustness of watermarking schemes against arbitrary attackers, and the formalization of those results in Coq. We used the Alea library, which formalizes probability theory and models probabilistic programs using a simple monadic translation. This work illustrates the strengths and particularities of the induced style of reasoning about probabilistic programs. Our technique for proving robustness is adapted from methods commonly used for cryptographic protocols, and we discuss its relevance to the field of watermarking.

1 Introduction

Watermarking consists in embedding some information inside a document in a *robust* and usually *imperceptible* way. It is notably used in digital property claims: a content owner may mark a document before its release, in order to be able to recognise copies of it and claim ownership. Of course, this cannot be done in a reliable and convincing way unless properties of the watermarking scheme have been solidly established.

A wide literature is dedicated to techniques for marking sound, still images and videos [7] that are robust against common transformations such as scaling, resampling, cropping, etc. This is achieved by using meaningful notions of distance and performing watermarking in an adequate space, such as frequency spectra. One may also consider documents as unstructured bit strings, and measure distortions using the Hamming distance, *i.*e., the number of positions where bits differ. In this setting, simple watermarking techniques may be used. The *bit-flipping* scheme involves a secret mask K which is a bitstring of the same length as the original document that is to be marked. Marking a document O (called *support* in this context) is done by changing the value of bits at positions set in K

* Work partially supported by SCALP project ANR-07-SESU-010 and grant 10-092309 from the Danish Council for Strategic Research.

— we say a position is set when the corresponding bit is 1. This can be expressed as a bitwise xor: $\texttt{mark}(K, O) = O \oplus K$. The induced distortion is the number k of bits set in K; this quantity is typically chosen to be a small fraction of the size of the support, and its value is assumed to be public. Detection of suspect copies O', written $\texttt{detect}(O, K, O')$, is done by counting the number of positions set in K where O and O' differ — in other words, the number of positions set in K where O' coincides with $\texttt{mark}(K, O)$. If that number exceeds a predefined threshold, the suspect document O' is claimed to be a copy of $\texttt{mark}(K, O)$. The idea behind this scheme is that an attacker who does not know K has little chance of changing enough bits for evading detection unless it also distorts the document so much that it renders it worthless. A variant of this technique is *substitution* watermarking where, instead of flipping bits at chosen positions in the original support, a secret message is used as a substitution for those bits. It is important to point out that although working with unstructured bitstrings is idealized, it has practical counterparts. For instance, the Agrawal, Haas and Kiernan's method [2] for watermarking numerical databases hides a mark by applying a substitution to least significant bits, and quality measurement is proportional to the Hamming distance.

Watermarked documents may be subject to attacks by malevolent users. A protocol is said to be *robust* against a particular attack if the mark can still be detected with high probability in copies modified by this attack. As observed above, it is useless to consider attacks that introduce excessive distortion: though such attacks are usually easy, they result in valueless data. Numerous attack scenarios can be envisioned, and it is impossible to test them all one by one. Nevertheless, we expect a protocol to be *provably robust*, *i.*e., robust against any attack. Provable security for watermarking is difficult to achieve, and there is currently little work in that direction. One reason for this is that watermarking protocols may be attacked at various levels, sometimes independently of the nature of documents and the marking algorithms. For example, in the context of digital property, someone may apply his own mark to marked data in order to claim ownership. To address such issues separately, Hopper, Molnar and Wagner [9] have introduced a distinction between strong watermarking schemes and non-removable embeddings (NREs). The former refers to the general protocol used to claim ownership, while the latter is the core technique for marking documents, consisting of a marking and a detection algorithm. Using cryptography and trusted third parties, they have formally proved (on paper) that strong watermarking protocols, robust against arbitrary attackers, can be derived from NREs. However, they did not address the robustness of particular embedding techniques, which depends on the kind of document that is considered. More generally, there is surprisingly little literature on robustness against arbitrary attackers, even in the idealised setting of bitstring algorithms. The problem has been identified, and several authors have proposed definitions and methodologies [1, 11], but they did not provide proofs of their results even for simple schemes. To the best of our knowledge, the only such proof has been carried out in the context of graph watermarking [12]. This proof relies on complex assumptions, concerning in particular the limited knowledge of the attacker, which may be hard to formalise. In contrast, our approach is very elementary, relying on a clear formalisation and basic combinatorics. We also note that, although a central piece in that proof of robustness for graphs is the (proved) robustness of a technique for marking integer vectors, this technique is not suitable for marking bitstrings under the Hamming distance, since it applies a modification to each component of the vector — which also eases considerably the robustness argument.

In this paper, we make two important steps towards provably secure watermarking protocols. First, we present a new proof of robustness against arbitrary attackers, which applies to bit-flipping and substitution watermarking for bitstrings. The key idea here is an efficient reduction of robustness to secrecy of the key K, which is used to obtain a tight enough bound on the probability of successful attacks. Second, we have fully formalised these proofs using the Coq proof assistant and the Alea library for reasoning about probabilities and probabilistic programs. Therefore, we provide a solid methodology for further work in the area. The formalisation of our work is also interesting in that it involves aspects such as information, secret and probabilistic algorithms which are challenging for the formal methods community. There are indeed few significant formalisations of randomised programs, although this field is receiving increasing attention [3, 8, 10]. Our work is most closely related to Certicrypt [4, 5], a framework built on top of Alea for formalising proofs of cryptographic protocols. Those proofs proceed by program transformations, reducing cryptographic games (represented using a deep embeding approach) to simpler ones corresponding to mathematical assumptions on cryptographic primitives, *e.g.*, the computational hardness of discrete logarithms in cyclic groups. As we shall see, our work on watermarking does not necessitate a model as complex as the one used in Certicrypt. But the main difference lies in the nature of our proofs: while our argument also revolves around a reduction, it mainly relies on concrete computations on bitstrings. The end result is also quite different: our proof is self-contained and does not use assumptions about complex mathematical primitives. Therefore, our Coq development provides an interesting case study for Alea, showing how elements of computational information theory may be carried out in that framework.

The rest of the paper is structured as follows. In Section 2 we motivate high-level modelling choices and present the Alea library which is used to formally realise these choices in Coq. Then, we describe our robustness proof in Section 3, detailing the main steps of the reduction for bit-flipping watermarking, and showing how it applies to substitution watermarking as well. We conclude by discussing our results and directions for future work in Section 4.

2 Formalisation

Security properties are commonly expressed as games played by the implementation of a service against an attacker. This methodology has the advantage of providing a clear and concrete characterisation of the flaw we want to avoid. The security of NREs has been defined in this way [9], and we shall formally define robustness as a variant of it. We assume that $\texttt{dist}(O, O')$ represents the distance between O and O' and that $\texttt{detect}(O, K, O')$ is true when O' is a marked copy of O with key K. An attacker $\mathcal{A}$ will break the robustness of a watermarking scheme when the following game answers true, *i*.e., the attacker produced an object close to the original and the mark is no longer detected:

```
$O' \leftarrow \texttt{mark}(K, O)$
$O'' \leftarrow \mathcal{A}(O')$
$\texttt{dist}(O', O'') < \gamma \wedge \neg\texttt{detect}(O, K, O'')$
```

Of course, attacking watermarking schemes in this game is usually going to be easy unless we assume that the attacker does not know K and O. Other parameters affect the difficulty and meaning of our game, most notably the degree of abstraction of the model and the assumptions on the computational resources of the attacker. These parameters have to be investigated in order to obtain a meaningful formalisation.

Symbolic models, where secrets are perfect and cannot be guessed, can be used to reveal security breaches in some protocols. However, their high level of abstraction would trivialise the study of watermarking schemes. We are interested in quantifying how hard it is for an attacker to remove a mark, potentially relying on random choices and leaked information. In order to do this, we place ourselves in a probabilistic computational model where secrets are regular data that has been randomly chosen and may therefore be guessed by another party. In that context, our robustness game can be rephrased as follows, where all computations are probabilistic:

```
$O \leftarrow \texttt{random\_support}()$
$K \leftarrow \texttt{random\_key}()$
$O' \leftarrow \texttt{mark}(K, O)$
$O'' \leftarrow \mathcal{A}(O')$
$\texttt{dist}(O', O'') < \gamma \wedge \neg\texttt{detect}(O, K, O'')$
```

The computational resources available to the attacker are also a very important parameter of security games. In the case of cryptography, security often relies on the (supposed) hardness of a few problems: for example, inverting a logarithm is essentially as hard as guessing its argument. Under such circumstances, it is possible for an attacker to succeed if it is allowed to compute long enough, but security can be preserved if the time needed is exponential in a parameter that can be chosen at will, typically the size of the secret key. The case of watermarking differs a lot. In that context, the size of the key is limited by the support, and does not offer significant control on the computational cost of an attack. But it does not matter because this cost is in fact irrelevant: unlike in cryptography, the attacker has (usually) no way to know if he succeeded, which makes brute force attacks impossible. Therefore, we only need to prove that the attacker has a low probability of removing the mark in one attempt, and we can do so without any constraint on the time or space available to the attacker.

As we have seen, formalising robustness requires the notions of probability and probabilistic program. Since we do not need to make restrictions on the use of resources by attackers, they can also be modelled using the same notion of probabilistic algorithm. Thus, we chose to carry out our formalisation in the Coq library Alea, since it provides exactly those fundamental ingredients. Note, however, that there is no essential reason why our work could not be done with another proof assistant.

2.1 Alea

Alea is a Coq library containing a formalisation of complete partial orders, of the unit interval `U` ($[0;1]$), of probability theory, and tools for reasoning about probabilistic programs. We quickly recall some of the key concepts used in Alea. More details can be found in [3].

A standard, deterministic program of type A evaluates to a single value of type A. A probabilistic program may evaluate to different values in different runs, defining a probability distribution over A. This is the approach taken in Alea, which provides machinery for building such distributions in a functional programming style, using a monadic interpretation.

A probabilistic program of type A is thus viewed as a distribution, *i*.e., a Coq object of type `distr` A. The distribution may also be called a measure, or experiment. It is something that one can integrate over: it roughly has the type $(A \to [0;1]) \to [0;1]$, taking a weight function over A and returning its sum over the measure. The integral $\int f\ dP$ is written `mu P f` in Coq (we also use the notation $\mu\ P\ f$). Given a property Q on A, let 1_Q be the function which is 1 when Q holds and 0 otherwise, then $\mu\ P\ 1_Q$ represents the probability that the output of program P satisfies Q. The distribution P comes with some properties, *e*.g., it must be guaranteed to preserve addition, $\mu\ P\ (f+g) = (\mu\ P\ f)+(\mu\ P\ g)$.

The basic programming constructs for randomised computation include:

- **return** a which given $a : A$, returns the Dirac's distribution at point a;
- **let** $x = d_1$ **in** d_2 which given a probabilistic program d_1 of type `distr` A and a probabilistic program d_2 of type `distr` B depending on x of type A, evaluates d_1, bind the resulting value to x and then evaluates d_2.

As an example of Alea, we shall show how to encode probabilistic Turing machines. We need a set Σ of states and a set A of letters. A predicate `accept` of type $\Sigma \to$ `bool` distinguishes accepting states. The output of a transition will be a record $\{$`next` $: \Sigma;$ `write` $: A;$ `move` $:$ `dir`$\}$ with `dir` the type of directions with two values `L` and `R`. A transition is a function `trans` which takes a state and a letter as arguments and gives an output. The configuration of the machine is given by a state, a tape and the current position of the tape, it is represented by a record $\{$`cur` $: \Sigma;$ `tape` $:$ `Z` $\to$ `A`$;$ `pos` $:$ `Z`$\}$.

It is trivial to define a deterministic function `update` which given a configuration and an output produces the new configuration. The Turing machine itself works as a program which recursively applies transitions and stops when it reaches an accepting state:

```
let rec run m = if accept (cur m) then return m
 else let o = trans (cur m) (tape m (pos m)) in run (update m o)
```

For the deterministic case, we cannot define this function in general in Coq because there is no way to guarantee the termination of the fixpoint. In Alea however, the type `distr` A includes sub-probabilities ($\mu\ P\ 1 < 1$) which models possibly non terminating programs. This type is equipped with a complete partial order structure and thus any monotonic continuous operator of type $(A \to$ `distr` $B) \to A \to$ `distr` B has a fixpoint. In order to model probabilistic Turing machines, we just have to see the transition function as a probabilistic transformation: `trans` : $\Sigma \to A \to$ `distr output` and interpret the previous definition of `run` as a function of type `config` $\to$ `distr config` using the monadic constructions for `return` and `let` and the general fixpoint on distributions. We can then measure the probability for the machine to terminate from an initial configuration m_0: it is given by the expression μ (`run` m_0) 1.

In the development of the watermarking algorithms, we only define simple probabilistic programs using structural recursion. However, the robustness of a watermarking scheme will be established using a quantification over all possible attackers. It is important to make sure that our formalisation covers everybody. The main restriction of our model is that random primitives are limited to discrete distributions. However this is sufficient for modelling probabilistic Turing machines, because we only need to express a distribution on outputs which is a finite set. Thus, our formalisation in Coq is adequate: any probabilistic attack can be expressed in our framework, and is thus covered by our results.

2.2 Reasoning with Alea

We shall describe a simple proof, establishing that some function implements the uniform probability law on bit vectors of a given length. We define the `BVrandom` function which given a natural number n, computes uniformly a bit vector of size n. It is defined by structural induction on n, using the `Flip` program which returns `true` with probability $\frac{1}{2}$ and `false` otherwise:

```
Fixpoint BVrandom n : distr (Bvector n) :=
  match n with
  | 0    ⇒ return Bnil
  | Sm ⇒ let hd = Flip in let tl = BVrandom m in return (Vcons hd tl)
  end
```

Then we seek to show that `BVrandom` implements a uniform distribution which means that the probability that `BVrandom` n returns a particular vector x of length n is 2^{-n}. Formally, it is written as follows where `BVeq` is a boolean function testing the equality of two bit vectors and `B2U` is the conversion of booleans to 0 and 1 in the set `U`.

```
Theorem BVrandom_eq : ∀n (x : Bvector n),
  μ (BVrandom n) (fun y ⇒ B2U (BVeq y x)) == (1/2)^n.
```

The interpretation of randomised programs as measures gives us a direct way to establish such a result by using simple algebraic transformations following the structure of the program. Examples of transformations are:

- μ (**return** x) $f == f\ x$,
- μ `Flip` $f == (1/2) \times f$ `true` $+ (1/2) \times f$ `false`,
- μ (**let** $x = P$ **in** $Q\ x$) $f == \mu\ P$ (**fun** $x \Rightarrow \mu\ (Q\ x)\ f$)

The `BVrandom` function is defined by structural induction on n, thus we naturally reason on it by induction on n or (preferably) on the vector x. The base case is trivial: `BVrandom` 0 computes to **return** `Bnil` and both sides simplify to 1, so we can invoke `reflexivity`. When the list has a head and tail, we unfold the computation of `BVrandom (S` n) and simplify the obtained measure to make μ (`BVrandom` n) f (almost) appear as:

$(1/2) \times \mu$ (`BVrandom` n) (**fun** $y \Rightarrow$ `B2U (BVeq (Vcons true` y) (`Vcons` $a\ x$)))+
$(1/2) \times \mu$ (`BVrandom` n) (**fun** $y \Rightarrow$ `B2U (BVeq (Vcons false` y)(`Vcons` $a\ x$)))
$== (1/2)^{\mathtt{S}\,n}$

We can conclude by doing a case analysis to compare the heads of those lists, followed by the invocation of the induction hypothesis, some simplifications (notably killing the branch where a is not the correct head) and finally we obtain equalities on measures and on reals in `U` which are solved automatically.

In the rest of the paper, we omit most occurrences of `B2U`, implicitly treat decidable propositions (*e*.g., equality on bit vectors) as booleans, and we simply write $\mathcal{P}(e)$ for $\mu\ e$ `B2U`, which represents the probability that e returns `true` when e is a probabilistic program computing a boolean, *i*.e., a Coq object of type `distr bool`.

3 Proof of Robustness

We shall now present our results. They have been fully formalised using Coq version 8.3, the AAC plugin [6] for reasoning modulo associativity commutativity, and version 7 of the ALEA library which benefited from several contributions as part of this work. The full development can be downloaded or replayed online at `http://www.lix.polytechnique.fr/~dbaelde/watermarking/`.

Our proof follows the main language-based technique for proving properties against arbitrary attackers: we reduce robustness to the simpler property of secrecy of the key. In the following, we detail the main steps of our proof for the bit flipping scheme: we first quantify secrecy, then use an efficient reduction of robustness to secrecy, and finally show that the obtained bound is exponentially small in the size of the support. For each step, we provide the informal argument and discuss how this argument carries to the Coq formalisation. Finally, we observe that the substitution algorithm can be treated in the exact same fashion.

3.1 Operations on Bit Vector

Since our algorithms rely heavily on bit vectors, we had to develop libraries for common bit vector operations, both deterministic and probabilistic. This part of our formalisation is mostly straightforward. It benefited a lot from reasoning modulo associativity and commutativity, and we also used type classes to reflect the boolean algebra structure that bit vectors inherit from booleans.

In this paper, we shall use the following notations, which depart slightly from our Coq notations. If X and Y are bit vectors of same length, $X \oplus Y$, $X \mathbin{\&} Y$ and $X \parallel Y$ respectively denote bit-wise xor, conjunction and disjunction, and $\neg X$ is the bit-wise negation of X. The number of bits set to 1 in X is $|X|_1$. The Hamming distance $\texttt{dist}\ X\ Y$ is defined as $|X \oplus Y|_1$. We also define $|X \setminus Y|_1$ as $|X \mathbin{\&} \neg Y|_1$: it counts the number of bits set in X but not in Y.

We defined a couple of uniform random generators: `BVrandom` n returns a bit vector of length n; `BVrandom_k` k n returns a vector of length n with k bits set; `BVrandom_k_mask` k M returns a vector of the same length as the mask M, having k bits set, only at positions where M is also set. This last function is used to randomly modify a vector, in a uniform way that is interesting to consider more closely.

Lemma 1 **(`RandomBV.correct_eq`).** *We define a correction function that takes a vector X, removes i bits among those set in X and adds m among those not set.*

$$\texttt{correct}\ X\ i\ m := \begin{cases} \textbf{let}\ I = \texttt{BVrandom\_k\_mask}\ i\ X\ \textbf{in} \\ \textbf{let}\ M = \texttt{BVrandom\_k\_mask}\ m\ (\neg X)\ \textbf{in} \\ \quad \textbf{return}\ ((X \mathbin{\&} \neg I) \parallel M) \end{cases}$$

For any bit vectors K and K' of length n, `correct` K' $|K' \setminus K|_1$ $|K \setminus K'|_1$ *returns K with the following probability:*

$$\binom{|K'|_1}{|K' \setminus K|_1}^{-1} \binom{n - |K'|_1}{|K \setminus K'|_1}^{-1}$$

Proof. Our lemma simply says that choices of I and M in the correction are independent, and thus the probability of success is the product of the probabilities of finding the two right vectors. However, the corresponding Coq proof turns out to be quite interesting. The key point is that we do not use conditional probabilities to express independence, but rely instead on the structure of the distribution (that is, the probabilistic program) to make independence explicit.

After unfolding a few constructs, and writing i for $|K' \setminus K|_1$ and m for $|K \setminus K'|_1$, the probability that we are considering is the following:

$$\begin{aligned} &\mu\ (\texttt{BVrandom\_k\_mask}\ i\ K')\ (\lambda I. \\ &\quad \mu\ (\texttt{BVrandom\_k\_mask}\ m\ (\neg K'))\ (\lambda M. \\ &\qquad K = (K' \mathbin{\&} \neg I) \parallel M)) \end{aligned}$$

One can show that `BVrandom_k_mask` k X always ranges among vectors Y such that $|Y \setminus X|_1 = 0$. This can be used (twice) to enrich our expression:

$$\begin{aligned} &\mu\ (\texttt{BVrandom\_k\_mask}\ i\ K')\ (\lambda I.\ (|I \setminus K'|_1 = 0) \times \\ &\quad \mu\ (\texttt{BVrandom\_k\_mask}\ m\ (\neg K'))\ (\lambda M.\ (|M \setminus (\neg K')|_1 = 0) \times \\ &\qquad (K = (K' \mathbin{\&} \neg I) \parallel M))) \end{aligned}$$

By the stability of distributions under scalar multiplication, we can push multiplications under μ constructs, at which point we observe that $(|I \setminus K'|_1 = 0) \wedge (|M \setminus (\neg K')|_1 = 0) \wedge (K = (K' \,\&\, \neg I) \parallel M)$ is equivalent to $(I = (K' \,\&\, \neg K)) \wedge (M = (K \,\&\, \neg K'))$. This relies on basic boolean algebra, proved notably by observing that for any two vectors u and v of size n, $u \,\&\, v = 0$ implies $u \,\&\, \neg v = u$. Thus, we have simplified our probability distribution as follows:

$$\begin{array}{l}\mu\ (\texttt{BVrandom\_k\_mask}\ i\ K')\ (\lambda I.\\ \quad \mu\ (\texttt{BVrandom\_k\_mask}\ m\ (\neg K'))\ (\lambda M.\\ \quad\quad (I = (K' \,\&\, \neg K)) \times (M = (K \,\&\, \neg K'))))\end{array}$$

In other words, we have succeeded in splitting the result of our computation in two independent parts, one for each probabilistic variable. Using the stability under multiplication in the opposite direction as before, we can thus split the whole computation in two:

$$\begin{array}{rl}& \big(\mu\ (\texttt{BVrandom\_k\_mask}\ i\ K')\ (\lambda I.\ (I = (K' \,\&\, \neg K)))\big)\\ \times & \big(\mu\ (\texttt{BVrandom\_k\_mask}\ m\ (\neg K'))\ (\lambda M.\ (M = (K \,\&\, \neg K')))\big)\end{array}$$

From there, it is easy to conclude by uniformity of our choice primitive.

3.2 Bit Flipping Watermarking

We now define the bit flipping watermarking scheme and its two security properties. Our definitions rely on a few parameters, which are considered public: n is the size of the mask and support; k is the number of marked bits; δ is the detection threshold, *i.*e., a message is considered marked if it has more than δ marked bits; γ is the deformation threshold, *i.*e., two messages are considered indistinguishable if their Hamming distance is less than γ. Obviously, those four parameters should be strictly positive integers. More assumptions will be introduced after the definitions.

For bit flipping on supports of size n, the key is a mask with only k bits set to 1, marking is done by flipping the bits set in the mask, and the number of marked bits in O' is the number of positions set in K where O' and O differ.

$$\begin{aligned}\texttt{genkey} &:= \texttt{BVrandom\_k}\ n\ k\\ \texttt{mark}\ K\ O &:= K \oplus O\\ \#\texttt{marks}\ O\ K\ O' &:= |K \,\&(O \oplus O')|_1\end{aligned}$$

We then give games defining robustness and secrecy. Robustness is the ability to resist unmarking challenges, where the attacker is given a marked message and has to return an unmarked message that is indistinguishable from the marked one:

$$\texttt{unmark}\ \mathcal{A} := \begin{cases}\textbf{let}\ K = \texttt{genkey}\ \textbf{in}\\ \textbf{let}\ O = \texttt{BVrandom}\ n\ \textbf{in}\\ \textbf{let}\ O' = \texttt{mark}\ K\ O\ \textbf{in}\\ \textbf{let}\ O'' = \mathcal{A}\ O'\ \textbf{in}\\ \quad \textbf{return}\ (\texttt{dist}\ O'\ O'' < \gamma\ \ \&\ \ \#\texttt{marks}\ O\ K\ O'' < \delta)\end{cases}$$

Secrecy is the impossibility for an attacker to guess the key, given only a marked message:

$$\texttt{find\_mask}\ \mathcal{A} := \begin{cases} \textbf{let}\ K = \texttt{genkey}\ \textbf{in} \\ \textbf{let}\ O = \texttt{BVrandom}\ n\ \textbf{in} \\ \textbf{let}\ O' = \texttt{mark}\ K\ O\ \textbf{in} \\ \textbf{let}\ K' = \mathcal{A}\ O'\ \textbf{in} \\ \quad \textbf{return}\ (K = K') \end{cases}$$

Finally, we make the following natural assumptions on our parameters:

$$\delta \leq k \quad \wedge \quad \gamma \leq n \quad \wedge \quad k \leq \gamma + \delta \quad \wedge \quad 2k \leq n \tag{1}$$

The first two require that the detection threshold is less than the number of marked bits, and that the deformation threshold is less than the support size. Values of δ and γ outside these ranges would be meaningless; another way to put it is that bounding them by k and n respectively does not change the problem at all. The third equation is more interesting: it states that an attack is possible. Indeed a successful attack must alter more than $k - \delta$ bits. Finally, the last inequality states that less than half of the bits should be marked. This is obviously desirable for a watermarking application. In any case, if more than half of the bits are marked, the problem is more simply attacked by trying to find the unmarked bits and we are back to a situation where the constraint is satisfied.

3.3 Secrecy

We prove that the key does not leak through marking: the knowledge of marked messages does not allow a better attack than the blind attack consisting of guessing the key from scratch. While the proof of this first lemma would be deemed straightforward on paper, its formalisation reveals a few interesting steps.

Lemma 2 (`Xor_analysis.secrecy`). *Guessing the key given a marked support is as hard as a blind guess:*

$$\forall \mathcal{A},\ \mathcal{P}(\texttt{find\_mask}\ \mathcal{A}) \leq \binom{n}{k}^{-1}$$

Proof. By definition, `find_mask` $\mathcal{A}$ is:

$$\begin{aligned} &\mu\ \texttt{genkey}\ (\lambda K.\ \mu\ (\texttt{BVrandom}\ n)\ (\lambda O.\ \mu\ (K \oplus O)\ (\lambda O'. \\ &\quad \mu\ (\mathcal{A}\ O')\ (\lambda K'.\ \textbf{return}\ (K = K'))))) \end{aligned}$$

The key observation is that for any vector K, the distribution $K \oplus \texttt{BVrandom}\ n$ is equal to `BVrandom` n. This is proved in `RandomBV.BVxor_noise` by induction on the size of vectors and straightforward computation on the distributions. Using it, we can rewrite `find_mask` $\mathcal{A}$ as follows:

$$\begin{aligned} &\mu\ \texttt{genkey}\ (\lambda K.\ \mu\ (\texttt{BVrandom}\ n)\ (\lambda O'. \\ &\quad \mu\ (\mathcal{A}\ O')\ (\lambda K'.\ \textbf{return}\ (K = K')))) \end{aligned}$$

From there, the proof is routine: we simply need to permute the various choices to make it explicit that $\mathcal{A}$ is blindly guessing the outcome of the uniform choice `genkey`. In ALEA, the permutation of probabilistic `let`-definitions corresponds to commuting integrals, which is not a trivially valid operation in general. However, it has been proved (in ALEA) to always be admissible for discrete distributions, and our distributions are indeed discrete because their domain itself is so. So we permute the introduction of K after that of O' and K', and use the uniformity of `genkey`, that is `BVrandom_k`:

$$\begin{pmatrix} \mu\ (\texttt{BVrandom}\ n)\ (\lambda O'. \\ \mu\ (\mathcal{A}\ O')\ (\lambda K'. \\ \quad \mu\ \texttt{genkey}\ (\lambda K.\ K = K'))) \end{pmatrix} \leq \begin{pmatrix} \mu\ (\texttt{BVrandom}\ n)\ (\lambda O'. \\ \mu\ (\mathcal{A}\ O')\ (\lambda K'. \\ \quad 1/\binom{n}{k})) \end{pmatrix}$$

From there, we conclude easily by simplifying the right hand-side program into $1/\binom{n}{k}$ multiplied by the probabilities that `BVrandom` n and $\mathcal{A}\ O'$ terminate.

3.4 Robustness

We now proceed to reduce robustness to secrecy. Given an attacker that can unmark messages, we build an attack on the key that runs the attack on the mark, considers the changes as an estimation of the key, and randomly attempts to correct it in order to obtain the secret key. In order to do this efficiently, we first bound the error on the estimated key, and then show that it suffices to guess the error to know how many bits are incorrectly set and how many bits are missing in the estimated key.

For the next two lemmas, we consider a run of an attack on the mark. Let K be a mask, *i*.e., a vector of size n satisfying $|K|_1 = k$. Let O be an arbitrary vector of size n, and O' be `mark` $K\ O$. Let O'' be another vector of size n, representing the attackers' attempt at removing the mark, and let $k_a =$ `dist` $O'\ O''$ be the number of attacked bits. We define K' to be $O' \oplus O''$. From the viewpoint of considering this set of changes as an approximation of K, we define the error e to be `dist` $K\ K'$, the number of incorrect bits k_i to be $|K' \setminus K|_1$ (these are the bits set to 1 in K' but not in K, they have no influence on mark detection but affect distortion) and the number of missing bits k_m to be $|K \setminus K'|_1$ (these are the bits which are 0 in K' but 1 in K so where the mark is not removed). We finally define the number of correct bits k_c to be $k - k_m$ (these are the bits set to 1 in both K and K').

Lemma 3 (Maximum error $e_{\mathbf{max}}$). *If the attack succeeds,* i.*e.,* `dist` $O'\ O'' < \gamma$ *and* $\#$`marks` $O\ K\ O'' < \delta$*, then* `dist` $K\ K' < e_{max}$ *where* $e_{max} := \gamma + 2\delta - k$.

Proof. The hypothesis $\#$`marks` $O\ K\ O'' < \delta$ gives us $k_m < \delta$. We also observe that `dist` $K\ K' = k_i + k_m$, and $k_i + k_c = |K'|_1 =$ `dist` $O'\ O'' < \gamma$. We conclude that `dist` $K\ K' = (k_i + k_c) - k_c + k_m = (k_i + k_c) + 2k_m - k < \gamma + 2\delta - k$.

In our COQ development, this reasoning is done directly in the proof that the reduction is correct, since it is quite simple, relying only on simple properties of boolean operations and linear arithmetic facts, which are proved automatically using the `omega` tactic of COQ.

For example, with $\gamma = k$ and $\delta = k/2$, we have $e_{\text{max}} = k$. This is a tight bound, in the sense that it can be reached, but it does not bring much information about K: in general, guessing K from K' with $\texttt{dist}\ K\ K' < k$ is essentially as hard as guessing from scratch the positions of the k bits set in K. Fortunately, we can extract much more information, as the next lemma shows.

Lemma 4 **(`Xor_analysis.ki_km`).** *The quantities k_i and k_m can be derived from k_a and e, since we have $k_i = \frac{e+k_a-k}{2}$ and $k_m = \frac{e+k-k_a}{2}$.*

Proof. Follows immediately from $k_a = k_i + k_c = k_i + (k - k_m)$ and $e = k_i + k_m$.

Definition 1 **(Reduction).** *Given an attack $\mathcal{A}$ on the mark, we build an attack on the key by starting with the estimated key K' corresponding to the attack on the mark, guessing randomly an error $0 \leq e < e_{max}$, deducing k_i and k_m, and guessing the correction of K' according to those parameters.*

$$\mathcal{A}'\ \mathcal{A}\ O' \overset{def}{=} \begin{cases} \textbf{let } O'' = \mathcal{A}\ O' \textbf{ in} \\ \textbf{let } K' = O' \oplus O'' \textbf{ in} \\ \textbf{let } k_a = \texttt{dist}\ O'\ O'' \textbf{ in} \\ \textbf{let } e = \texttt{random\_int}\ e_{max} \textbf{ in} \\ \textbf{let } k_i = (e + k_a - k)/2 \textbf{ in} \\ \textbf{let } k_m = (e + k - k_a)/2 \textbf{ in} \\ \quad \textbf{return } (\texttt{correct}\ K'\ k_i\ k_m) \end{cases}$$

Note that $\mathcal{A}'$ can be enhanced to not consider values of e which yield non-integer k_i and k_m. We do not care to do it here since this would not change asymptotic results.

Lemma 5 **(`Xor_analysis.reduction`).**

$$\forall \mathcal{A}.\ \mathcal{P}(\texttt{unmark}\ \mathcal{A}) \leq e_{max} \times \binom{\gamma}{\lceil \gamma/2 \rceil} \times \binom{n-k+\delta}{\delta} \times \mathcal{P}(\texttt{find\_mask}\ (\mathcal{A}'\ \mathcal{A}))$$

Proof. We consider an execution, introducing O, K, O', then executing $\mathcal{A}'\ \mathcal{A}\ O'$. With probability $\mathcal{P}(\texttt{unmark}\ \mathcal{A})$ we'll obtain an unmarked copy O'' from $\mathcal{A}\ O'$. Then, the correct error $\texttt{dist}\ K\ K'$ will be guessed with a probability of one in e_{max}, and the correct values for k_i and k_c will follow from it. Finally, $\texttt{correct}\ K'\ k_i\ k_m$ will return K with a probability of one in

$$\binom{k_a}{k_i} \times \binom{n-k_a}{k_m}$$

It only remains to bound those two terms independently of k_i, k_m and k_a, using monotonicity properties of the combinatorial function and assumptions on our parameters and on the success of the unmarking attack.

Theorem 1 **(`Xor_analysis.robustness`).**

$$\forall \mathcal{A}.\ \mathcal{P}(\texttt{unmark}\ A) \leq \frac{e_{max} \times \binom{\gamma}{\lfloor \gamma/2 \rfloor} \times \binom{n-k+\delta}{\delta}}{\binom{n}{k}}$$

Proof. Immediate from Lemmas 2 and 5.

3.5 Asymptotic Behaviour

We now show that the bound derived in Theorem 1 is negligible, *i.*e., that it is eventually bounded by a negative exponential. This standard practice with security bounds is less relevant in watermarking than, for example, in cryptography, because the size of our secret key is fixed. Thus, a precise expression of the bound may be more useful in practice than an asymptotic over-approximation. Nevertheless, addressing the latter is an interesting challenge in terms of formal proofs. In order to achieve this, we move from $[0;1]$ to positive real numbers, which are better behaved and notably form a semi-ring. Specifically, we use a new experimental formalisation of $\mathbb{R}^+$ built on top of `U` in which a real number is represented as a pair with an integral part in `nat` and a fractional part in `U`.

In the following, we are going to study the behaviour of our bound for n large enough. To do so, we need some information on how other parameters grow with n. We shall essentially assume that those parameters are linear in n, which is the standard choice in practice. In addition to (1), we make the following assumptions for some $0 < \alpha < 1$, $c \in \mathbb{N}$ and $e, g \in \mathbb{R}^+$ with non-null c and g:

$$k/n \leq \alpha \quad (2) \qquad\qquad e\text{max} \leq e \times n \quad (4)$$

$$(n/k)^c \geq 4 \quad (3) \qquad\qquad k - \delta \geq n \times g + \lfloor \gamma/2 \rfloor \times c \quad (5)$$

The last assumption is formulated in an ad-hoc fashion, but it essentially requires that the difficulty of the attack is large enough and growing linearly with n. Indeed, it quantifies the gap between the maximum number of bits that can be attacked (γ) and the minimum that has to be changed in order to evade detection ($k - \delta$).

When all parameters are linear, we can always determine α, c and e. However, a suitable value for g in (5) can only be found when $k - \delta > \lfloor \gamma/2 \rfloor \times c$. For example, suppose we want to mark $k = \lfloor n/100 \rfloor$ bits with a detection threshold $\delta = \lfloor n/200 \rfloor$. We can take $c = \log(4)/\log(100)$, and assumption (5) roughly requires that γ is less than six times $k - \delta$. Thus, our hypotheses still allow us to cover a satisfyingly wide range of attacks. Our asymptotic bound will become tighter when the attack is made more difficult, *i.*e., γ is made smaller relative to $k - \delta$.

Lemma 6 (`Asymptotic.final`). *There exists $\beta \in \mathbb{R}^{+*}$ and $m \in \mathbb{N}$ such that for any $n \geq m$,*

$$\frac{e\mathit{max} \times \binom{\gamma}{\lfloor \gamma/2 \rfloor} \times \binom{n-k+\delta}{\delta}}{\binom{n}{k}} \leq \alpha^{\lfloor \beta n \rfloor}$$

Proof. We first show that:

$$e\text{max} \times \binom{\gamma}{\lfloor \gamma/2 \rfloor} \times \frac{\binom{n-k+\delta}{\delta}}{\binom{n}{k}} \leq e\text{max} \times (2 \times 4^{\lfloor \gamma/2 \rfloor}) \times (\frac{k}{n})^{k-\delta}$$

This is proved by approximating separately each term. For the second term, we use the following precise bound: $\forall k, \binom{2k}{k} \leq 4^k$. For the third term, we show that

$$\frac{\binom{n-k+\delta}{\delta}}{\binom{n}{k}} = \frac{\delta + 1 \times \ldots \times k}{(n-k+\delta+1) \times \ldots \times n} = \frac{\delta+1}{n-k+\delta+1} \times \frac{\delta+2}{n-k+\delta+2} \times \ldots \times \frac{k}{n}$$

and we conclude by observing that we have $k - \delta$ increasing factors.

We choose β to be $g/2$, and m is chosen to be large enough so that $2 \times e_{\max} \leq \alpha^{-\lfloor gn/2 \rfloor}$, which can always be obtained since $e_{\max}$ is only linear in n by (4). We obtain the following bound: $\alpha^{-\lfloor gn/2 \rfloor} \times 4^{\lfloor \gamma/2 \rfloor} \times (\frac{k}{n})^{k-\delta}$. By assumption (3) we can further enlarge this into $\alpha^{-\lfloor gn/2 \rfloor} \times (k/n)^{-c\lfloor \gamma/2 \rfloor} \times (k/n)^{k-\delta}$. Using (5) we obtain $\alpha^{-\lfloor gn/2 \rfloor} \times (k/n)^{\lfloor ng \rfloor}$ and we finally obtain $\alpha^{\lfloor gn/2 \rfloor}$ by (2).

Corollary 1 (`Asymptotic.robustness`)**.** *Provided that assumptions (1-5) hold, there exists $\beta > 0$ and $m \in \mathbb{N}$ such that for any $n \geq m$,*

$$\forall \mathcal{A}.\ \mathcal{P}(\texttt{unmark}\ A) \leq \alpha^{\lfloor \beta n \rfloor}$$

3.6 Substitution Watermarking

We now consider marking by substitution. In this scheme, the secret key is made of two parts: a mask M and a message K, both of the same length as the support to be marked. Instead of flipping bits according to the mask as in the previous scheme, bits of the support are replaced by those of the message at positions indicated by the mask. Although the message contains irrelevant information (at positions not set in the mask) in this presentation, this is not a problem for our development.

$$\begin{aligned}
\texttt{genkey} &:= \begin{cases} \textbf{let}\ M = \texttt{BVrandom\_k}\ n\ k\ \textbf{in} \\ \textbf{let}\ K = \texttt{BVrandom}\ n\ \textbf{in} \\ \quad \textbf{return}\ (M, K) \end{cases} \\
\texttt{mark}\ (M, K)\ O &:= (O \,\&\, \neg M) \,\|\, (K \,\&\, M) \\
\#\texttt{marks}\ (M, K)\ O' &:= |M \,\&\, \neg(K \oplus O')|_1
\end{aligned}$$

Note that this scheme is *blind*, *i.*e., the detection method does not need to refer to the initial support O as before. This property is the reason why substitution is preferred to bit flipping in practice. Indeed, blind schemes allow to delegate the search of marked copies to several agents without having to communicate much information to those agents: When the secret key is generated using a pseudo-random number generator (PRNG) it suffices to communicate the seed that was used to initialise it. Of course, this compression advantage disappears in our theoretical setting; we discuss this in conclusion.

The definition of the robustness game `unmark` is the same as before, and it turns out that we can derive the exact same bound as for bit-flipping watermarking, under the same hypotheses on parameters n, k, δ and γ. The reason is that although there is more information in the secret key, it suffices to guess the mask M to obtain the (useful part of) the message K. Therefore the problem is exactly the same as for bit flipping, the same reduction applies and we obtain the same bound. Formally, the only change is in the definition of the secrecy game:

$$\texttt{find\_mask}\ \mathcal{A} := \begin{cases} \textbf{let}\ (M,K) = \texttt{genkey}\ \textbf{in} \\ \textbf{let}\ O = \texttt{BVrandom}\ n\ \textbf{in} \\ \textbf{let}\ O' = \texttt{mark}\ (M,K)\ O\ \textbf{in} \\ \textbf{let}\ M' = \mathcal{A}\ O'\ \textbf{in} \\ \quad \textbf{return}\ (M = M') \end{cases}$$

Lemma 7 **(`Substitution_analysis.secrecy`).**

$$\forall \mathcal{A},\ \mathcal{P}(\texttt{find\_mask}\ \mathcal{A}) \leq \binom{n}{k}^{-1}$$

Lemma 8 **(`Substitution_analysis.reduction`).** *Using $\mathcal{A}'$ in Definition 1:*

$$\forall \mathcal{A}.\ \mathcal{P}(\texttt{unmark}\ A) \leq e_{max} \times \binom{\gamma}{\lceil \gamma/2 \rceil} \times \binom{n-k+\delta}{\delta} \times \mathcal{P}(\texttt{find\_mask}\ (\mathcal{A}'\ \mathcal{A}))$$

4 Conclusion

We have given a robustness result for bit-flipping and substitution schemes against arbitrary attackers, based on a new reduction to secrecy. This reduction and the associated proofs have been fully formalised in Coq on top of the Alea library. Our work is one of the few large examples of using this library, illustrating most of its key aspects with the notable exception of non-terminating probabilistic functions and fixpoints of (sub)distributions. In itself, our development totals around 900 lines of specification and 1500 lines of proofs. But it also required further developments of Alea, including properties of binomial coefficients, a few dedicated tactics to simplify expressions involving distributions, the integration of material about discrete distributions adapted from Certicrypt and the development of the library on positive real numbers (1100 lines of spec and 1700 lines of proofs). Ignoring all the infrastructure work, the core of our development is satisfyingly concise: each of the two robustness proof has only about 200 lines of proofs that follow quite closely the informal presentation.

Our work could be pushed further in several directions. Now that basic watermarking primitives have been formalised and proved robust, one could consider formalising and proving more complex schemes and protocols. For instance, one could prove the security of complete watermarking protocols built from robust NREs. In order to formalise existing proofs in that domain [9], on needs notions such as oracles and polynomial time attackers, and therefore we would naturally turn to the more complex framework of Certicrypt. But there are also fundamental questions which remain open even at the level of the basic embedding primitives. In some contexts, it is relevant to consider repeated attacks, and one should study the advantage that attackers can gain by having access to multiple supports marked with the same key, or multiple marked copies of the same support with different keys. A related question will be to study the impact on robustness of non-uniform distributions for supports, messages, and of other distortion measures than Hamming distances. For example, in the context of watermarking numerical databases, the meaningful notion of distance is based on query answers, which opens up the possibility of *subsetting attacks* that consist in removing a few lines or columns from the database. There are techniques to prevent such attacks, that could be formalised in our framework. Finally, a very important question is the issue of pseudo-random number generators. To the best of our knowledge, there is no formal security work that takes pseudo-random generators into account. This is not a problem in many applications, since true randomness becomes increasingly accessible from physical devices. However, as discussed in the substitution watermarking section, pseudo-random generators are used as a compression device in watermarking. Evaluating the security impact of this practice is thus unavoidable, and promises to be very difficult.

References

1. André Adelsbach, Stefan Katzenbeisser, and Ahmad-Reza Sadeghi. A computational model for watermark robustness. In *Proceedings of the 8th international conference on Information hiding*, IH'06, pages 145–160, Berlin, Heidelberg, 2007. Springer-Verlag.
2. Rakesh Agrawal, Peter J. Haas, and Jerry Kiernan. Watermarking Relational Data: Framework, Algorithms and Analysis. *VLDB J.*, 12(2):157–169, 2003.
3. Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. *Science of Computer Programming*, 74(8):568–589, 2009. A preliminary version appeared in the proc. of MPC 2006.
4. Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In Zhong Shao and Benjamin C. Pierce, editors, *POPL*, pages 90–101. ACM, 2009.
5. Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 71–90. Springer, 2011.
6. Thomas Braibant and Damien Pous. Tactics for reasoning modulo AC in Coq. In *Proc. 1st CPP*, volume 7086 of *LNCS*, pages 167–182. Springer, 2011.
7. Ingemar J. Cox, Matthew L. Miller, and Jeffrey A. Bloom. *Digital Watermarking*. Morgan Kaufmann Publishers, Inc., San Francisco, 2001.
8. Osman Hasan and Sofiene Tahar. *Probabilistic Analysis Using Theorem Proving*. VDM Verlag, 2008.
9. Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In *Proceedings of the 4th conference on Theory of cryptography*, TCC'07, pages 362–382, Berlin, Heidelberg, 2007. Springer-Verlag.
10. Joe Hurd. *Formal Verification of Probabilistic Algorithms*. PhD thesis, University of Cambridge, 2002.
11. Stefan Katzenbeisser. A computational model for digital watermarks. in Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2005), 2005.
12. Sanjeev Khanna and Francis Zane. Watermarking maps: hiding information in structured data. In *SODA*, pages 596–605, 2000.