Browser Randomisation against Fingerprinting: A Quantitative Information Flow Approach

Frédéric Besson, Nataliia Bielova, Thomas Jensen

Browser Randomisation against Fingerprinting: a Quantitative Information Flow Approach

Frédéric Besson, Nataliia Bielova, and Thomas Jensen*

Inria, France

Abstract. Web tracking companies use device fingerprinting to distinguish the users of the websites by checking the numerous properties of their machines and web browsers. One way to protect the users' privacy is to make them switch between different machine and browser configurations. We propose a formalisation of this privacy enforcement mechanism. We use information-theoretic channels to model the knowledge of the tracker and the fingerprinting program, and show how to synthesise a randomisation mechanism that defines the distribution of configurations for each user. This mechanism provides a strong guarantee of *privacy* (the probability of identifying the user is bounded by a given threshold) while maximising *usability* (the user switches to other configurations rarely). To find an optimal solution, we express the enforcement problem of randomisation by a linear program. We investigate and compare several approaches to randomisation and find that more efficient privacy enforcement would often provide lower usability. Finally, we relax the requirement of knowing the fingerprinting program in advance, by proposing a randomisation mechanism that guarantees privacy for an arbitrary program.

1 Introduction

Web tracking companies are actively using device fingerprinting to identify the users of the websites by checking the numerous properties of their machines and web browsers. While this technique is of great value to trackers, it is a threat to users' privacy. The Panopticlick project [10] was the first to demonstrate the power of fingerprinting, while recent research shows that this technique is widely used by web tracking companies [1, 18]. Today, only few solutions exist for protecting the users from being fingerprinted. Acar *et al.* [1] have analysed these solutions and concluded that none of them can guarantee user privacy. For example, the Firegloves [5] browser extension returns randomised values when queried for certain browser attributes. However since the same attributes can be retrieved via different browser APIs, the users of Firegloves become more uniquely identifiable than users who do not install this extension. Nevertheless, the idea of such randomisation is a promising approach to counter fingerprinting

* This research was partially supported by the French ANR-10-LABX-07-01 Laboratoire d'excellence CominLabs.

but its foundations should be developed further. In this paper, we propose a theory of privacy enforcement by randomisation and show *what privacy guarantee can be achieved*. From this theory, we derive an enforcement mechanism for obtaining this guarantee.

Example 1. For a simple illustration, consider the distribution of the browser names and the potentially fingerprinting program `P1` from Fig. 1. The distribution of browser names is known to the tracker and is called an *a priori distribution*, and a concrete program output transforms it into an *a posteriori distribution* that we show for `o = B`. Assuming there are 50 visitors to the website,

name	p(name)
`Firefox`	0.49
`Chrome`	0.49
`Opera`	0.02

```
1 if (name = "Opera")
2 then o := A;
3 else o := B;
4 output o;
```

name	p(name|B)
`Firefox`	0.5
`Chrome`	0.5
`Opera`	0.0

Fig. 1: Pre-distribution, program `P1`, post-distribution after observing B.

only one will have an Opera browser, and hence will be uniquely identified by executing the program. Notice that other 49 visitors are indistinguishable since the execution of the program will yield an output `o = B` for all of them.

Inspired by Clarkson *et al.*'s work on belief revision [6], Mardziel *et al.* [16,17] propose a definition of *knowledge threshold security* stating that a program is secure if all the post-beliefs of all possible secrets are bounded by some threshold t. Espinoza and Smith [11] discuss this definition and name it *worst-case posterior vulnerability* underlining that it is very biased towards the worst output. In order to enforce *knowledge threshold security*, Mardziel *et al.* [16,17] suggest to: i) run the program if the threshold holds for all the values of the secret input; ii) not run the program in case there is at least one value for which the guarantee does not hold. This radical approach forbids all the safe users to run the program. Typically, the program of Fig. 1 would not run because the single Opera user would be identified whereas 98% of the users could run the program safely.

In this paper, we show how to enforce *knowledge threshold security* using a more flexible mechanism based on randomisation. For example, given 50 website visitors, program `P` will be evaluated to provide a worst-case probability of guessing the identity equal to 1 for Opera users and $\frac{1}{49}$ for Firefox and Chrome users. Then, for a threshold $t = \frac{1}{25}$, we will provide a mechanism that randomises the browser name for Opera users, and not influence the experience of other users.

1.1 Attacker model and assumptions

We consider terminating (deterministic or probabilistic) programs operating over a finite range of inputs. Upon termination, the program returns a single value.

As we only consider terminating programs, the attacker will always observe a value. In our model, the attacker has arbitrary computing power, he provides the program and observes the output. The input of the program is secret and represents a browser configuration. However, the attacker has perfect knowledge over the distribution of the secret inputs. Our enforcement works by randomising the program inputs. We consider that the attacker has access to the precise description of the randomisation mechanism.

The main contributions can be summarised as:

- A model of the problem of privacy protection against fingerprinting programs, based on information-theoretic channels representing the statistics of browser properties and the program.
- A novel definition of privacy for such systems, that ensures that the probability of an attacker identifying a user is bounded by a given threshold. We show that the enforcement of privacy can be achieved by randomising the browser properties, and that this randomisation problem can be reduced to solving linear programs.
- Algorithms (a global, a greedy and a decentralised) for enforcing privacy against a particular fingerprinting program. All algorithms ensure the strong privacy guarantee, *i.e.*, that the probability of being identified is smaller than a given threshold. The algorithms optimise the solution with respect to additional "usability" constraints, which ensure that randomisation is used as little as possible.
- A general result about how user privacy can be guaranteed for any program that the user might run. This represents the worst case scenario, in which a program checks all the possible browser properties of the user. This result is important in the case where it is difficult or impossible to construct the information-theoretic channel that models the program (*e.g.*, due to the complexity of a language such as JavaScript).

The paper is organised as follows. In Section 2, we show how to model fingerprinting in information-theoretic terms. In particular, we define t-privacy which is our formal notion of privacy. We also formally introduce the problem of enforcing t-privacy. In Section 3, we show that the enforcement problem reduces to solving a Linear Program. In Section 4, we present several enforcement algorithms that trade optimality for efficiency. In Section 5, we propose an enforcement that ensures t-privacy for any program. Related works are discussed in Section 6 and Section 7 concludes. More details, in particular proofs of theorems, can be found in a companion report [3].

2 Threshold-based privacy and usability for fingerprinting

This section shows how to model fingerprinting in terms of information-theoretic channels [7]. The input to the fingerprinting script is the browser configuration that is the user's secret. A script P can be modelled by an information-theoretic

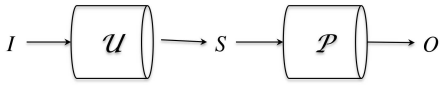

Fig. 2: Cascade of channels $\mathcal{U}$ and $\mathcal{P}$.

channel $\mathcal{P} = (S, O, P)$ which produces an output $o \in O$ given a secret configuration $s \in S$. The input/output transformation is given by a matrix P such that $P[s, o]$ is the conditional probability $p(s|o)$ of observing the output o given the secret input s. For deterministic scripts, the corresponding channel can be extracted by running the script for all the possible inputs. These inputs are the possible browser configurations whose distribution is known to the attacker. For probabilistic scripts, probabilistic sampling would construct a reliable but approximate channel. Even better, we can construct an exact channel using symbolic computations over probability distributions. For each possible input, we can evaluate the script semantics expressed as a distribution transformer [6]. Symmetrically, we can run a weakest pre-expectation calculus [12] which extends weakest precondition calculus to probabilistic programs. The model acquisition problem is not in the scope of this paper, and henceforth we just assume that a channel matrix is given.

User identities are related to the browser configurations by browser statistics. We model this mapping by a deterministic channel $\mathcal{U} = (I, S, U)$, where I is a finite set of user identities, S is a finite set of possible browser configurations and U is a channel matrix, where

$$U[i, s] = \begin{cases} 1 \text{ if user } i \text{ has configuration } s \\ 0 \text{ otherwise.} \end{cases}$$

By construction the matrix U is deterministic, meaning that each row contains only one entry equal to 1, and the rest are 0s. In other words, $U[i, s]$ means that a user i possesses only one configuration s. For a deterministic channel $\mathcal{C}$, we write $Im(C, i)$ for the unique o such that $C[i, o] = 1$ and $Pre(C, o)$ for the set of inputs that can produce o: $Im(C, i) = o$ iff $C[i, o] = 1$ and $Pre(C, o) = \{i | C[i, o] = 1\}$.

Initially, all the users are equally indistinguishable and therefore the initial attacker knowledge of user identities is modelled by the uniform distribution. The worst-case probability of guessing a user identity by observing a run of the cascade of channels $\mathcal{U} \otimes \mathcal{P}$ (Fig. 2) is given by

$$\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) = \max_{i \in I, o \in O} p(i|o) = \max_{i \in I, o \in O} \frac{\sum_{s \in S} U[i, s] \cdot P[s, o]}{\sum_{i'} \sum_{s \in S} U[i', s] \cdot P[s, o]}.$$

This result specialises the definition of worst-case a posteriori distribution for the a priori uniform distribution and a sequence of channels [3, Section 4.1]. Suppose that this quantity equals a threshold t then for all user i the probability of being identified by an attacker observing a run of the fingerprinting script is below t. Definition 1 formalises this notion of threshold-based privacy.

Definition 1 (Threshold-based privacy). *A channel $\mathcal{C}$ is* t-private *if the probability of guessing the channel's input is bounded by t:* $\mathbb{P}(\mathcal{C}) \leq t$.

Note that a fingerprinting script only runs once per session of a given user. Therefore an attacker cannot accumulate knowledge by observing several runs of several fingerprinting scripts. In other words, different runs of fingerprinting scripts cannot be correlated and the maximum fingerprinting capability of a script is adequately modelled by the quantity $\mathbb{P}(\mathcal{U} \otimes \mathcal{P})$.

Example 2. Consider the program `P1` from Example 1. Fig. 3 shows the matrix of a user channel $\mathcal{U}$ representing (simplified) browser statistics and the matrix of a channel $\mathcal{P}$ for program `P1`. When a user with identity i_5 runs the program

U	`Firefox`	`Opera`
i_1	1	0
i_2	1	0
i_3	1	0
i_4	1	0
i_5	0	1

P	`A`	`B`
`Firefox`	0	1
`Opera`	1	0

Fig. 3: User channel and channel for program `P1`.

`P`, a tracker observes an output `A` and knows that the user has the configuration Opera. The channel $\mathcal{U}$ then makes the user i_5 uniquely identifiable, therefore the a posteriori probability is $p(i_5|\mathtt{A}) = \frac{U[i_5,\mathtt{Opera}]\cdot P[\mathtt{Opera},\mathtt{A}]}{U[i_5,\mathtt{Opera}]\cdot P[\mathtt{Opera},\mathtt{A}]} = 1$. When an attacker observes output `B`, he concludes that the user has a configuration different from Opera. The channel $\mathcal{U}$ makes the users different from i_5 indistinguishable for the attacker thanks to the program output `B`. Therefore for all these users $p(i|\mathtt{B}) = \frac{1}{4}$ since output `B` can be obtained from Firefox configurations. We conclude that the worst-case probability of guessing the user identity is:

$$\mathbb{P}(\mathcal{U} \otimes \mathcal{P}) = \max_{i \in I}\{p(i|\mathtt{A}); p(i|\mathtt{B})\} = \max\{1; 1/4\} = 1.$$

In the following, we propose enforcement mechanisms which replace the channel $\mathcal{U}$ with a randomised channel $\mathcal{R}$ so that the cascade $\mathcal{R} \otimes \mathcal{P}$ is t-private for a given threshold t. The enforcement will also minimise the randomisation characterised by a quantity $\mathbb{U}(\mathcal{R},\mathcal{U})$ (see Def. 3). However, not every threshold can be enforced by user channel randomisation. We state our enforceability theorem as follows.

Definition 2 (Enforceable threshold). *A threshold t is* enforceable *for a user channel $\mathcal{U}$ and program channel $\mathcal{P}$ if there exists a randomised channel $\mathcal{R}$, such that $\mathcal{R} \otimes \mathcal{P}$ is t-private ($\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$).*

Theorem 1 (Enforceability). *A threshold t is enforceable for a user channel $\mathcal{U} = (I, S, U)$ and any program channel $\mathcal{P}$ if and only if $\frac{1}{|I|} \leq t$.*

Proof. See companion report [3, Theorem 3]. □

Our enforcement mechanism does not pick an arbitrary randomised channel $\mathcal{R}$ but aims at maximising the informal usability requirement that "the users do not want to switch to other configurations too often since they prefer to use their original configurations as much as possible". Formally, this requirement is captured by the following definition.

Definition 3 (Usability). *Given a channel $\mathcal{U}$, the usability of a channel $\mathcal{R}$ is* $\mathbb{U}(\mathcal{R}, \mathcal{U}) = \sum_i R[i, Im(U, i)]$.

Usability quantifies the amount of switching of configurations incurred by a given channel $\mathcal{R}$. We aim at maximising usability *i.e.*, minimising the probability that a user needs to switch between configurations. The maximum theoretical usability is obviously obtained when $\mathcal{R} = \mathcal{U}$ and $\mathbb{U}(\mathcal{U}, \mathcal{U}) = |I|$.

For the program `P1`, we can graphically represent the space of possible randomisation channels $\mathcal{R}$ that enforce t-privacy, where all Firefox users would get a probability x for Firefox and $1 - x$ for Opera, dually Opera users would get probability y for using Opera and $1 - y$ for Firefox. The probability of guessing the user identity of a channel $\mathcal{R} \otimes \mathcal{P}$ is:

$$\max \left\{ \frac{x}{4x + 1 - y}, \frac{1 - x}{4 - 4x + y}, \frac{1 - y}{4x + 1 - y}, \frac{y}{4 - 4x + y} \right\}.$$

while the usability of a channel $\mathcal{R}$ is $\mathbb{U}(\mathcal{R}, \mathcal{U}) = 4x + y$. Therefore, the usability

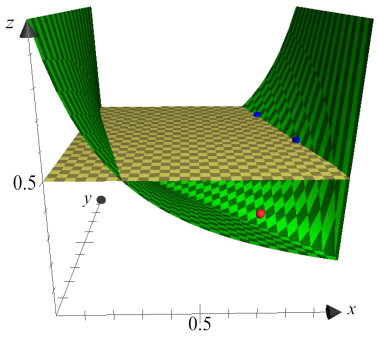

Fig. 4: The dark green surface represents the probability of guessing the user identity of a channel $\mathcal{R} \otimes \mathcal{P}$, while the yellow plane shows the probability of guessing the identity set to $\frac{1}{2}$. The blue points $x = 0.9, y = 0.4$ and $x = 0.8, y = 0.8$ are reaching the threshold t. The red point $x = 0.75$, $y = 0.3$ corresponds to the randomisation channel $\mathcal{R}_0$ of Fig. 5 and belongs to the surface below the threshold t.

of the channel $\mathcal{R}_0$ (red point) is $\mathbb{U}(\mathcal{R}_0, \mathcal{U}) = 4 \cdot 0.75 + 0.3 = 3.3$, while the usability of randomisation channels presented by blue points is $4 \cdot 0.9 + 0.4 = 4$ and $4 \cdot 0.8 + 0.8 = 4$ respectively. In the next section we show that the usability of 4 is the maximum possible usability of a randomisation channel that enforces $\frac{1}{2}$-privacy given the user channel $\mathcal{U}$ and the program `P1`.

3 Reduction to Linear Programming

This section shows how to construct a randomised channel $\mathcal{R}$ that ensures t-privacy for the composed channel $\mathcal{R} \otimes \mathcal{P}$. We parametrised the matrix of the channel $\mathcal{R}$ using the variables $R[i,s] = x_{is}$, where $i \in I, s \in S$ and define a system of constraints that such a channel must satisfy. Finding the best possible channel can then be expressed as a Linear Program (LP).

Channel correctness The channel matrix R represents conditional probabilities, therefore:

- $0 \leq x_{is} \leq 1$ for all $i \in I, s \in S$, meaning each parameter is a probability.
- $\sum_{s \in S} x_{is} = 1$ for all $i \in I$, meaning each matrix row is a distribution.

***t*-privacy** : the channel $\mathcal{R}$ must guarantee t-privacy for the composed program channel $\mathcal{P}$ *i.e.*, $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$. Expanding this expression, we get:

$$\max_{i \in I, o \in O} \frac{\sum_{s \in S} x_{is} \cdot P[s,o]}{\sum_{j \in I} \sum_{s \in S} x_{js} \cdot P[s,o]} \leq t.$$

This can be rewritten as a system of linear inequalities for all $i \in I, o \in O$:

$$\sum_{s \in S} x_{is} \cdot P[s,o] - t \cdot \sum_{j \in I} \sum_{s \in S} x_{js} \cdot P[s,o] \leq 0.$$

The system of constraints presented by the channel correctness and t-privacy requirements have a number of solutions. One solution is a channel $\mathcal{R}$ where all the matrix elements are equal to $\frac{1}{|S|}$. To see this, observe that 1) the channel is correct : $\sum_{s \in S} \frac{1}{|S|} = 1$; 2) t-privacy is guaranteed for any $\mathcal{P}$ since $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq \mathbb{P}(\mathcal{R})$ (see [3, Theorem 2]), and $\mathbb{P}(\mathcal{R}) = \frac{1/|S|}{\sum_{j \in I} 1/|S|} = \frac{1}{|I|} \leq t$ for any enforceable t (Theorem 1). However, this solution might not guarantee the best usability: it forces each user to use other configurations as often as his original configuration.

This last observation motivates a third requirement:

Usability Our usability requirement (see Definition 3) exactly describes the desire of users to switch to other configurations as rarely as possible. Therefore, the usability of a randomised channel $\mathcal{R}$ must be maximised. Remember that usability $\mathbb{U}(\mathcal{R}, \mathcal{U}) = \sum_{i \in I} R[i, Im(U,i)]$ represents a sum of all entries in R where $U[i,s] = 1$. Therefore, we can rewrite it as a requirement to maximise the function $\sum_{(i \in I): U[i,s]=1} x_{is}$.

Combining the requirements presented in this section, we can state our problem as a *Linear Program*, where the usability must be maximised while the channel correctness and t-privacy constraints are satisfied:

$$max \ \sum_{(i \in I): U[i,s]=1} x_{is} \text{ s.t.}$$
$$\begin{cases} 0 \leq x_{is} \leq 1 & \forall i \in I, s \in S \\ \sum_{s \in S} x_{is} = 1 & \forall i \in I \\ \sum_{s \in S} x_{is} \cdot P[s,o] - t \cdot \sum_{j \in I} \sum_{s \in S} x_{js} \cdot P[s,o] \leq 0 & \forall i \in I, o \in O \end{cases}$$

Example 3. Consider again the program `P1` from Example 2. For a user channel $\mathcal{U} = (U, I, S)$, the randomised channel $\mathcal{R}$ has a matrix of $|I| \times |S|$ parameters denoted by x_{is} (see Fig. 5). The usability of $\mathcal{R}$ is computed as follows:

$$\mathbb{U}(\mathcal{R}, \mathcal{U}) = \sum_{(i \in I) : U[i,s]=1} x_{is} = x_{11} + x_{21} + x_{31} + x_{41} + x_{52}.$$

The constraints imposed by channel correctness are straightforward to write down. Here, we present the constraints provided by the t-privacy requirement, where the threshold $t = \frac{1}{2}$:

$$\begin{aligned}
&+\tfrac{1}{2}\, x_{11} - \tfrac{1}{2}\, x_{21} - \tfrac{1}{2}\, x_{31} - \tfrac{1}{2}\, x_{41} - \tfrac{1}{2}\, x_{51} \leq 0 \\
&-\tfrac{1}{2}\, x_{11} + \tfrac{1}{2}\, x_{21} - \tfrac{1}{2}\, x_{31} - \tfrac{1}{2}\, x_{41} - \tfrac{1}{2}\, x_{51} \leq 0 \\
&\ldots \\
&-\tfrac{1}{2}\, x_{12} - \tfrac{1}{2}\, x_{22} - \tfrac{1}{2}\, x_{32} + \tfrac{1}{2}\, x_{42} - \tfrac{1}{2}\, x_{52} \leq 0 \\
&-\tfrac{1}{2}\, x_{12} - \tfrac{1}{2}\, x_{22} - \tfrac{1}{2}\, x_{32} - \tfrac{1}{2}\, x_{42} + \tfrac{1}{2}\, x_{52} \leq 0
\end{aligned}$$

This LP has several solutions. In Fig. 5 we have the optimal channel $\mathcal{R}_1$ with usability $\mathbb{U}(\mathcal{R}_1, \mathcal{U}) = 4$ and the non-optimal channel $\mathcal{R}_0$ of Fig 4.

R	`Firefox`	`Opera`
i_1	x_{11}	x_{12}
i_2	x_{21}	x_{22}
i_3	x_{31}	x_{32}
i_4	x_{41}	x_{42}
i_5	x_{51}	x_{52}

R_0	`Firefox`	`Opera`
i_1	0.75	0.25
i_2	0.75	0.25
i_3	0.75	0.25
i_4	0.75	0.25
i_5	0.7	0.3

R_1	`Firefox`	`Opera`
i_1	0.9	0.1
i_2	0.8	0.2
i_3	0.7	0.3
i_4	0.6	0.4
i_5	0	1

Fig. 5: Parametrised channel $\mathcal{R}$ and randomised channels R_0 and R_1.

3.1 Feasibility

In practice, switching to a particular configuration might not be technically feasible. For instance, faithfully emulating a foreign OS might be considered too drastic a change. Other switchings of configurations could hinder an optimal user experience. For instance, switching to an arbitrary language could have undesirable side-effects. For personal reasons, certain users might also wish to keep their configuration unaltered. To model those feasibility constraints, we can introduce a relation $\mathit{Imp} \subseteq I \times S$ representing the configuration switches that are considered impossible. The fact that those switchings between configurations might be impossible can be encoded into the Linear Program in a straightforward way. Each pair $(i, s) \in \mathit{Imp}$ yields the constraint $x_{is} = 0$. These constraints can be used to pre-process the problem and substitute x_{is} for 0, thus reducing the number of variables of the problem.

4 Enforcement algorithms

Linear programs can be solved in polynomial time using interior points methods. Despite a worst-case exponential complexity, the Simplex algorithm is usually superior and, on average, only a linear number of pivot steps are needed. Furthermore, for probabilistic models, this average linear bound can be formally established [20]. Yet, the complexity is fairly high and solving the linear program directly might not be computationally feasible. In particular, the complexity is parametrised by the number of identities $|I|$ and does not exploit properties of the program channel. In the following, we show that it is possible to substantially reduce the number of parameters needed to construct an optimal channel $\mathcal{R}$.

Indistinguishable identities. A key insight is that two identities i and i' are indistinguishable for a script $\mathcal{P}$ as soon as they are mapped to the same secret configuration s by the channel $\mathcal{U}$ ($U[i] = U[i']$). In the companion report [3, Theorem 4 of Section 6.2], we prove that there exists an optimal randomised channel $\mathcal{R}$ such that indistinguishable identities get the same enforcement ($R[i] = R[i']$). We exploit this property to simplify the LP formulation and reduce the number of variables from $|I| \cdot |S|$ to $|S| \cdot |S|$ and the number of constraints from $|I| \cdot |O| + |I| \cdot |S| + |I|$ to $|S| \cdot |O| + |S| \cdot |S| + |S|$. This transformation has a drastic impact on the size of the resulting linear program which becomes independent from the number of identities.

Greedy algorithm We can exploit the structure of the program and reduce further the complexity at the cost of a potential non-optimal utility. In particular, we can detect identities i whose probability of guessing is below the threshold with the original channel $\mathcal{U}$. Definition 4 captures this notion.

Definition 4. *Given channels $\mathcal{P}$ and $\mathcal{U}$, an identity i is* locally safe *iff*

$$\frac{1}{|Eq_{\mathcal{U}\otimes\mathcal{P}}(i)|} \leq t \text{ where } Eq_{\mathcal{C}}(i) = \{i' | \forall o, C[i, o] = C[i', o]\}.$$

In other words, the probability of guessing of *locally safe* identities is below the threshold t whatever the mapping of other identities.

Theorem 2. *Let Safe be the locally safe identities for P w.r.t. $\mathcal{U}$. Then there is a channel $\mathcal{R}$ such that $\mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$ and $\forall i \in Safe : R[i, Im(U, i)] = 1$.*

Proof. Given an identity $i_0 \in Safe$, the channel $\mathcal{R}$ can be constructed as follows:

$$\forall i \in Safe, R[i, j] = \begin{cases} 1 \text{ if } j = Im(U, i) \\ 0 \text{ if } j \neq Im(U, i) \end{cases} \quad \forall i \notin Safe, R[i, j] = \begin{cases} 1 \text{ if } j = Im(U, i_0) \\ 0 \text{ if } j \neq Im(U, i_0) \end{cases}$$

All locally safe identities $i \in Safe$ are unmodified and therefore their probability of guessing is below the threshold t. All the other identities $i \notin Safe$ are modified so that they are undistinguishable from the safe identity i_0 and therefore their probability of guessing is also below the threshold t.

Using Theorem 2, we devise a greedy algorithm for solving the linear program. Identifying the locally safe identities can be done by performing the channel composition $\mathcal{U} \otimes \mathcal{P}$ and counting the number of identical lines. The remaining constraints can be solved using standard linear programming algorithms. The size of the LP has been further reduced: there are $|S| \cdot (|S| - |Safe|)$ variables and $(1 + |S|) \cdot (|S| - |Safe|) + |S| \cdot |O|$ constraints. This is a substantial gain: the number of variables is decreasing as a linear function of $|Safe|$. Moreover, even if the remaining constraints do not completely vanish, they are sparser.

This algorithm is maximising usability locally because *locally safe* identities are not modified. However, there is no guarantee that this strategy would provide the maximum usability.

Example 4. Consider a threshold $t = \frac{1}{2}$ and the enforcement of $\frac{1}{2}$-privacy for the channels $\mathcal{U}$ and program $\mathcal{P}$ given below:

```
1 if name="Firefox"
2 then o:=A
3 else{
4   if name="Opera"
5   then o:=B
6   else o:=C}
```

U	Firefox	Opera	Chrome
i_1	1	0	0
i_2	1	0	0
i_3	0	1	0
i_4	0	0	1

P	A	B	C
Firefox	1	0	0
Opera	0	1	0
Chrome	0	0	1

Note that identities i_1 and i_2 are indistinguishable. As the threshold is $\frac{1}{2}$, i_1 and i_2 are locally safe

$$Eq_{\mathcal{U} \otimes \mathcal{P}}(i_1) = Eq_{\mathcal{U} \otimes \mathcal{P}}(i_2) = \{i_1, i_2\}.$$

Hence, the *greedy algorithm* is solving the following parametrised $\mathcal{R}$ channel:

R	Firefox	Opera	Chrome
i_1	1	0	0
i_2	1	0	0
i_3	x_{20}	x_{21}	x_{22}
i_4	x_{30}	x_{31}	x_{32}

The best solution with $\mathbb{U}(\mathcal{R},\mathcal{U}) = 3$ is obtained for

$$x_{20} = x_{30} = 0 \qquad x_{21} = x_{22} = x_{31} = x_{32} = \tfrac{1}{2}.$$

Here, the identities i_1 and i_2 keep their secrets while i_3 and i_4 are uniformly randomised over Opera and Chrome.

Decentralised algorithm The previous algorithm can be modified to work in a completely decentralised fashion where each row of the channel R can be computed independently. If an identity i is *locally safe*, we get

$$R[i,j] = \begin{cases} 1 & \text{if } j = Im(U,i) \\ 0 & \text{otherwise} \end{cases}.$$

Otherwise, if an identity i' is not locally safe, it needs to switch configuration. This can be done by identifying an identity that is locally safe. If there is none, the identity i maximising $|Eq_{\mathcal{U} \otimes \mathcal{P}}(i)|$ can be chosen and therefore the identities are all indistinguishable.

Example 5. Continuing with the channels of Example. 4, the decentralised algorithm also allows identities i_1 and i_2 to keep their secrets unchanged. However, i_3 and i_4 would switch to the safe identity i_1. Compared to the greedy algorithm, the decentralised algorithm obtains a smaller usability : $\mathbb{U}(\mathcal{R},\mathcal{U}) = 2$.

For deterministic programs, a dynamic (or hybrid) information flow monitor [2] can evaluate if the knowledge contained in a program result is below the threshold. Using the decentralised algorithm, if the quantity of information is below the threshold, it is safe to output the result. If the identity is not proved locally safe by the dynamic monitor, the identity i maximising $|Eq_{\mathcal{U}\otimes\mathcal{P}}(i)|$ can be obtained using the previous algorithm. This identity i should be used instead of the original user identity. Returning a result $\perp$ distinct from existing outputs *i.e.,* an observable termination of the program does not achieve the same privacy guarantee. In practice, we can expect that there are many *locally* safe configurations that can be identified *e.g.*, by enumerating identities by decreasing popularity and running a hybrid monitor. Using this scheme, the usability might not be optimal but the computation is decentralised and the overhead small for identities for which local safety can be established by hybrid monitoring.

5 Enforcing t-privacy for any program

Finally, we consider the case when the program channel $\mathcal{P}$ is unknown or cannot be computed. In this case, only the user channel $\mathcal{U} = (I, S, U)$ is given, and we need to ensure that for any program $\mathcal{P}$, the probability of identifying the user is bounded by t.

5.1 Randomisation of the identity user channel $\mathcal{U}_{Id}$

Before providing a generic solution for any user channel, we first consider the simpler case where the user channel denotes a 1-1 mapping between user identities and configurations. Without loss of generality, we have that $I = S$ and we denote this channel by $\mathcal{U}_{Id} = (I, I, U_{Id})$, where U_{Id} is the identity matrix.

Like in previous sections, we find a randomised channel $\mathcal{R}_{Id} = (I, I, R_{Id})$ s.t. t-privacy is enforced while usability is maximised by solving a (simplified) linear program. Notice that the threshold t must be enforceable for a channel $\mathcal{U}_{Id}$ in the sense of Theorem 1.

$$max \ \sum x_{ii} \text{ s.t.}$$
$$\begin{cases} 0 \leq x_{ij} \leq 1 & \forall i, j \in I \\ \sum_{j\in I} x_{ij} = 1 & \forall i \in I \\ x_{ik} - t \cdot \sum_{j\in I} x_{jk} \leq 0 & \forall i, k \in I \end{cases}$$

This problem has the following solution:

$$R_{Id}[i,j] = \begin{cases} t & \text{if } i = j \\ \frac{1-t}{|I|-1} & \text{otherwise} \end{cases}$$

with the probability of guessing: $\mathbb{P}(\mathcal{R}_{Id}) = t$ and the usability: $\mathbb{U}(\mathcal{R}_{Id},\mathcal{U}_{Id}) = t \cdot |I|$. Interestingly, usability depends on the threshold: the higher the threshold t, the bigger is the probability that the user can be identified, hence his original configuration should change less, and therefore more usability would be provided.

5.2 Randomisation of an arbitrary channel $\mathcal{U}$

We now construct a randomised user channel $\mathcal{R}$ from a given user channel $\mathcal{U}$, that satisfies the t-privacy for any program channel $\mathcal{P}$: $\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$. Like before, a threshold t must be enforceable in a sense of Theorem 1.

We build channel $\mathcal{R}$ by starting from a channel $\mathcal{R}_{Id}$ and merging the columns of the identities that share the same configuration in the user channel $\mathcal{U}$.

$$R[i,s] = \begin{cases} t + \frac{1-t}{|I|-1} \cdot (n_s - 1) & \text{if } i \in Pre(U,s) \\ \frac{1-t}{|I|-1} \cdot n_s & \text{otherwise} \end{cases} \quad (1)$$

where $n_s = |Pre(U,s)|$ (the set Pre was defined in Section 2). We now prove the main properties of the constructed channel $\mathcal{R}$.

Lemma 1 (Well-formedness). *Given a user channel $\mathcal{U} = (I,S,U)$, a randomised channel $\mathcal{R} = (I,S,R)$, where R is computed by equation (1) is well-formed,* i.e.*,*

- $0 \leq R[i,s] \leq 1$ *for all $i \in I$, $s \in S$, meaning each parameter is a probability*
- $\sum_{s \in S} R[i,s] = 1$ *for all $i \in I$, meaning each matrix row is a distribution*

Lemma 2 (t-privacy). *Given a user channel $\mathcal{U}$, the randomised channel $\mathcal{R}$, where R is computed by equation (1) for an enforceable threshold t, is t-private:*

$$\mathbb{P}(\mathcal{R}) \leq t.$$

We can now prove the main theorem of this section: a constructed randomised user channel $\mathcal{R}$ can ensure t-privacy for any program channel $\mathcal{P}$.

Theorem 3. *For any user channel $\mathcal{U} = (I,S,U)$, the randomised user channel $\mathcal{R}$ ensures t-privacy for any program channel $\mathcal{P} = (S,O,P)$:*

$$\forall \mathcal{P} : \mathbb{P}(\mathcal{R} \otimes \mathcal{P}) \leq t$$

We do not prove that the randomised user channel $\mathcal{R}$ provides an optimal usability. The usability of the solution $\mathcal{R}$ given a user channel $\mathcal{U}$ is:

$$\mathbb{U}(\mathcal{R},\mathcal{U}) = \sum_{s \in S} |Pre(U,s)| \cdot (t + \frac{1-t}{|I|-1} \cdot (|Pre(U,s)| - 1)).$$

We now state the lower and upper bounds of the usability that is defined by the characteristics of the channel $\mathcal{U}$:

$$t \cdot |I| \leq l \cdot |S| \cdot (t + \frac{1-t}{|I|-1} \cdot (l-1)) \leq \mathbb{U}(\mathcal{R},\mathcal{U}) \leq h \cdot |S| \cdot (t + \frac{1-t}{|I|-1} \cdot (h-1)) \leq |I|$$

where $l = \min_{s \in S} Pre(U, s)$ and $h = \max_{s \in S} Pre(U, s)$.

In the general case, the randomised user channel $\mathcal{R}$ constructed in this section will provide a solution with reduced usability compared to the solutions provided by other approaches in previous sections. The reason for this is the fact that a program channel $\mathcal{P}$ may already make some of the configurations indistinguishable. For the users of such configurations, it is (in principle) possible to obtain a better usability.

6 Related work

Evaluation of information flow Mardziel *et al.* [16, 17] define the notion of *knowledge threshold secure* program. This is a generalisation of t-privacy allowing to attach different thresholds to different secrets. In our context, as we wish to protect privacy (and not secrets), only a single threshold is needed. Mardziel *et al.* are using an abstract domain of probabilistic polyhedra for computing an over-approximation of the threshold security of a program. They exploit this information to implement a simple enforcement algorithm: If the program is *threshold secure*, it runs without modification; if it is not *threshold secure*, it does not run at all. Our enforcement offers a better usability at the price of randomising the program inputs. Yet, our enforcement algorithms can ensure a minimum randomisation that is thus affecting a minimal set of users. For example, a greedy algorithm ensures that locally safe users will always run the program without any changes.

Klebanov [13, 14] has proposed efficient algorithms for exactly computing standard quantitative information flow measures of programs such as conditional (minimal) guessing entropy. The algorithms are either based on SAT-solving techniques [14] or on *extended Barvinok counting* [22]. These techniques are applied only to restricted classes of programs. SAT-based techniques require a propositional encoding of programs. Extended Barvinok counting consists in computing the number of integer points in a parametrised polyhedron and thus applies to programs that can be specified using linear integer arithmetic. In our theoretical setting, channels can model arbitrary terminating programs with a finite input/output relation but constructing explicitly the channel matrix could be costly. More efficient enforcement algorithms could certainly benefit from syntactic program restrictions. For deterministic programs, Klebanov's approaches can be adapted for deciding whether a program is t-private. Notice that Klebanov is only concerned with the problem of quantifying information flows and does not consider enforcement. However, this information can be directly exploited to implement the simple enforcement proposed by Mardziel *et al.*.

Köpf and Rybalchenko [15] approximate the entropy of a program using a combination of static and dynamic analyses, where random sampling is enhanced by a symbolic backward analysis. The method ensures a rapid convergence rate but the guarantee is probabilistic. Because t-privacy is a property quantifying over all the program inputs, it cannot be established by random sampling.

Moreover, the purpose of t-privacy is to protect all users (especially those whose configurations are rare).

***k*-anonymity** Sweeney [21] proposes *k-anonymity* that requires that every individual is anonymous within some set of at least size k. k-anonymity was not widely adopted as a privacy definition for two major reasons: 1) the values of sensitive attributes in the remaining set could be discovered due to their little diversity; 2) attackers with background knowledge of the distribution of the secret inputs can still infer some information about the secrets despite the fact that k-anonymity is enforced. The first problem related to k-anonymity is shown by an example when a program's output could have been caused by one of k possible inputs, but one of those inputs is much more probable than the rest. After observing a k-anonymous answer of a query, the a posteriori distribution of the secrets represents this knowledge of the attacker. The probability of guessing the secret given this knowledge is bounded by t thanks to t-privacy guarantee. The second problem is not applicable in our case since the attacker does not have any background knowledge: he collects all the data about the user through the execution of the program, and he has no history of interaction with the user because he cannot identify the user.

Differential privacy Dwork et al. [8] proposed a new privacy definition: a query to the database is *ϵ-differentially private* if and only if its answer is very similar to this query answer over a database that differs in only one record. In other words, one record in a database does not significantly change the answer of a query. Differential privacy was designed to reason about databases, while our probability of guessing is defined over guessing the only one secret: the user identity. Mardziel *et al.*, [17] make the observation that threshold based privacy and differential privacy can by formally compared using the notion of *ϵ-adversarial privacy* [19]. This notion was proven to be equivalent to differential privacy for a certain class of a priori distributions of input secrets (uniform distribution in our case). In our notations ϵ-adversarial privacy can be defined as follows.

Definition 5. *A channel $\mathcal{C} = (C, I, O)$ is* ϵ-adversarially private *iff for all input secrets $i \in I$ and for all output $o \in O$ we have $p(i|o) \leq e^{\epsilon} p(i)$.*

As we consider only one a priori distribution of secrets, *ϵ-adversarial privacy* definition coincides with our definition of *t-privacy* where $t = \frac{e^{\epsilon}}{|I|}$. Because differential privacy protects against a class of attackers the security guarantee is formally stronger. For this reason, algorithms for differential privacy [9] would therefore randomise scripts that are already t-private (but not *ϵ-adversarial private*) thus reducing their usability. In our fingerprinting context, we exploit the attacker's a priori knowledge for synthesising a channel that is t-private with minimal randomisation.

7 Conclusions and further work

Web tracking uses browser fingerprinting to identify users via scripts that obtain information about the browser's configuration. To protect users from such tracking, we propose a privacy enforcement mechanism based on randomisation of the script input. Our security guarantee is that the probability of guessing an identity by observing a script output is below a threshold t. We have presented a series of algorithms for enforcing t-privacy, all based on a Linear Programming formulation of the problem. The algorithms provide various trade-off between efficiency and *usability* where usability means that as little randomisation as possible is used. The exact resolution of the LP provides optimal usability. We also provide an enforcement mechanism ensuring the t-privacy of arbitrary programs at the cost of a reduced usability.

In our model, the attacker and the enforcement mechanism have perfect knowledge of the channel $\mathcal{U}$ *i.e.*, the distribution of the configurations is known. In a fingerprinting context, there are databases providing detailed information about the statistics of browser configurations [10] but a perfect knowledge of the distribution of the browser configuration worldwide is not realistic. As future work, we will investigate how to extend our framework to model partial knowledge *e.g.,* the fact that the user channel $\mathcal{U}$ belongs to a set $\mathbf{U}$.

One extension could *e.g.*, consist in synthesising a channel $\mathcal{R}$ that would ensure t-privacy with respect to the set $\mathbf{U}$. If $\mathbf{U}$ is expressed as a parametrised distribution, the enforcement problem can be stated as a non-linear optimisation problem instead of a Linear Program. Another extension consists in considering that the attacker might have an imprecise pre-belief [6] ($\mathcal{U} \in \mathbf{U}$) and ensure, for instance, that the channel is t-private with high probability. We could also consider that the attacker does not know the precise $\mathcal{R}$ channel but only the properties it enforces. In that case, the enforcement could ensure t-privacy at a better usability.

A longer-term extension of our model consists in modelling the dynamics of browser configurations. This dynamics is an obstacle to fingerprinting as fingerprints need to resist to modification of configurations. In theory, this should allow to design enforcement mechanisms providing a better usability. One of the author has started a preliminary practical evaluation of the evolution of fingerprints [4]. However, more research is needed to incorporate this knowledge into a formal model.

References

1. G. Acar, M. Juárez, N. Nikiforakis, C. Díaz, S. F. Gürses, F. Piessens, and B. Preneel. FPDetective: dusting the web for fingerprinters. In *CCS'13*, pages 1129–1140. ACM, 2013.
2. F. Besson, N. Bielova, and T. Jensen. Hybrid information flow monitoring against web tracking. In *CSF'13*, pages 240–254. IEEE, 2013.
3. F. Besson, N. Bielova, and T. Jensen. Enforcing browser anonymity with quantitative information flow. Technical Report 8532, Inria, 2014.

4. N. Bielova and P. Palladino. Stopfingerprinting, 2013. `https://stopfingerprinting.inria.fr/`.
5. K. Boda. Firegloves. `http://fingerprint.pet-portal.eu/?menu=6`.
6. M. R. Clarkson, A. C. Myers, and F. B. Schneider. Quantifying information flow with beliefs. *Journal of Computer Security*, 17(5):655–701, 2009.
7. T. M. Cover and J. A. Thomas. *Elements of Information Theory (2. ed.)*. Wiley, 2006.
8. C. Dwork. Differential privacy. In *ICALP*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.
9. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.
10. P. Eckersley. The Panopticlick project. `https://panopticlick.eff.org`.
11. B. Espinoza and G. Smith. Min-entropy as a resource. *Inf. Comp.*, 226:57–75, 2013.
12. F. Gretz, J.-P. Katoen, and A. McIver. Operational versus weakest precondition semantics for the probabilistic guarded command language. In *QEST*, pages 168–177. IEEE, 2012.
13. V. Klebanov. Precise quantitative information flow analysis - a symbolic approach. *Theor. Comput. Sci.*, 538:124–139, 2014.
14. V. Klebanov, N. Manthey, and C. Muise. SAT-based analysis and quantification of information flow in programs. In *QUEST*, volume 8054 of *LNCS*, pages 156–171. Springer, 2013.
15. B. Köpf and A. Rybalchenko. Approximation and randomization for quantitative information-flow analysis. In *CSF'10*, pages 3–14. IEEE, 2010.
16. P. Mardziel, S. Magill, M. Hicks, and M. Srivatsa. Dynamic Enforcement of Knowledge-based Security Policies. In *CSF'11*, pages 114–128. IEEE, 2011.
17. P. Mardziel, S. Magill, M. Hicks, and M. Srivatsa. Dynamic enforcement of knowledge-based security policies using probabilistic abstract interpretation. *Journal of Computer Security*, 21(4):463–532, 2013.
18. N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 541–555, 2013.
19. V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: Output perturbation for queries with joins. In *PODS'09*, pages 107–116. ACM, 2009.
20. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.
21. L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002.
22. S. Verdoolaege, R. Seghir, K. Beyls, V. Loechner, and M. Bruynooghe. Counting integer points in parametric polytopes using barvinok's rational functions. *Algorithmica*, 48(1):37–66, 2007.