Shuffling the Cards: An Information-Theoretic Defense Against Side Channel Attacks

***Abstract*—Side-channel attacks traditionally assume a landscape where a single process is observed in isolation. However, this leaves open a weakness in these attacks; if a second, decoy process is carefully introduced, the adversary may not know which process they are observing. First we introduce a formalization of side-channel leakage when decoy processes are present. Then we propose a novel defense against a broad class of side-channel attacks which bounds the information leaked in a side channel, allowing our defense to provably mitigate leakage, regardless of how the adversary processes their side-channel observations. To apply our method, we require that the side-channel leakage follows a particular pattern—small amounts of information leaked at known intervals. Our modifications to the algorithm being defended are minimal, only requiring the insertion of "breakpoints" in the algorithm. Finally, to show viability, we demonstrate how to adapt our defense to traffic analysis of streaming videos.**

I. Introduction

When designing cryptographic algorithms, cryptographers carefully choose the inputs and outputs of their algorithms such that adversaries cannot discover their secrets, such as cryptographic keys. However, when implementing these algorithms in the real world, the adversary may gather information from unexpected channels. For example, consider the following pseudocode:

```
1: secret ←$ {0,1}^128
2: for i in 1,...,100 do
3:     if secret[i] = 0 then
4:         Perform slow operation
5:     else if secret[i] = 1 then
6:         Perform fast operation
7:     end if
8:     Print i
9: end for
```

An adversary who interacts with the above program can't learn the value of the secret by looking at the prints directly. However, if the adversary watches the clock, they could guess the secret based on the time between print statements. This is called a timing attack, and this type of attack has been shown to be viable in practice [4]. Other channels include electromagnetic radiation [2], cache misses [10], and power usage [6]. Our example is not contrived: the timing channel leakage for the above program is very similar to the power channel leakage of RSA [5].

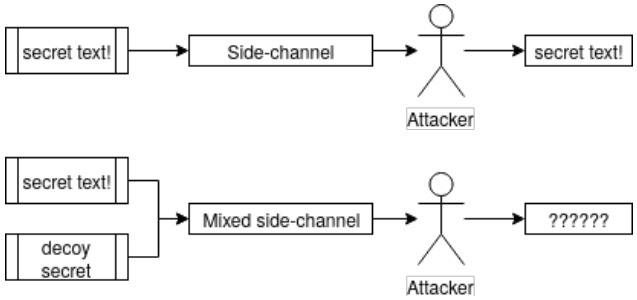

Fig. 1. Top: A simplified diagram of the traditional side-channel attack. Bottom: A simplified diagram of our defense.

Now consider the following change to the above program: initialize two instances, one real and one decoy. Start a random instance. Whenever a print happens, switch active instances with some probability. It now becomes much harder for the adversary to guess the secret. This is the main intuition behind our defense. We illustrate this idea in figure 1.

Building off this idea, we present a novel, provable defense against a wide variety of side channel attacks. Our technique is agnostic to the medium of transmission used in a side channel attack, and requires assumptions that are applicable to many side channels. Specifically, our guarantees hold under the following assumptions:

1) Side-channel information is leaked at discrete intervals (e.g. above, the time between printing)
2) The information leaked over each interval is small (e.g. above, either the first or second branch is taken; in other words "first" or "second" is the leakage)
3) No information is leaked over non-targeted side channels (because fixing a leaky faucet won't help if your pipe has burst)

At a high level our method works by taking a program,

breaking up the execution into steps, where the steps have some predefined explicit leakage, and then running multiple decoy copies of the same program, with different inputs. The executions of the programs are then carefully **interlaced** using the methods described in sections III, IV, such that the information leaked overall is reduced as a result of the interlacing of the leaked bits of information. We can then subsequently bound the fraction of secret information that will be leaked.

In order to measure the security of our defence, we take an information-theoretic approach. We describe the maximum amount of information the adversary learns from the side channel, measured by entropy, as defined by Shannon [14]. In particular, we model our defense as a noisy channel, where the adversary's view is perturbed due to the decoy instances. We first develop a formal model for side-channels under interlaced processes in section II. Then we present our defenses and their bounds in sections III, IV. We derive the maximum information the adversary can learn as the channel capacity of that channel.

We then demonstrate a case study in section V, where we show how to apply our algorithm to the domain of video fingerprinting [13], which is a form of side channel attack. We show how our theoretical model maps neatly onto this domain, and what the implementation of our system would theoretically look like.

Our technique does have some limitations. Our technique incurs the performance overhead of running the decoy processes, which cannot be parallelized. For adequate information-theoretic performance, our technique requires the execution steps to leak small amounts. This means channels whose leakage draws from a large set, for example the the cache miss channel, are not applicable. On the other handle, channels whose leakage draws from a small set, for example a power channel leaking the chosen branch, are good candidates for our technique.

II. Formalization of Side Channels

We extend the work of Köpf *et al.* [7] in formalizing side channel attacks. In Köpf *et al.*'s model, the side channel is defined as a function $f : K \times M \to S$ where in a traditional cryptosystem side channel, K is the set of secret keys, M is the set of messages, and S[1] is the set of side channel observations by the adversary. In a more general sense, one can think of $\mathsf{sk} \in K$ as the secret state and $\mathsf{msg} \in M$ as the public state.

We extend this model in two ways. First, we add a notion of **discretization**, where the output of the side channel can be broken into a time-series *emission list*. Second, we introduce a notion to formalize the observed output when multiple instances are allowed to run on the same processor. We call the schedule by which the instances are ordered an **interlace**. A strong notion of discretization allows us to reason about information-theoretic bounds on interlaces. Discretization and interlacing act as a layer on top of the model of Köpf *et al.* – specifically, any defense that is bounded under Köpf *et al.*'s model can only see improved bounds under our model, as we show towards the end of this section.

[1]In Köpf *et al.*, O is used. We use O for other purposes in this work, so we use S here instead.

Discretization can be described by dividing any $f(\mathsf{sk}, \mathsf{msg})$ into a series of discrete observations, which we call **emissions**. Formally, let ST be the set of program states and E be the set of emissions. After discretization, a function f becomes a list of functions $\bar{g} = (g_1, ..., g_l)$, $g_i : ST \to ST \times E$, where $l = \mathsf{len}_f(\mathsf{sk}, \mathsf{msg})$. The input for g_1 is $st = (\mathsf{sk}, \mathsf{msg})$. The input for any other g_i is the state output from g_{i-1}. We call each g_i an **interval**.

A $\bar{g}$ is a valid **discretization** of f if the concatenation of all emission outputs from executing $g_1, ..., g_l$ is identical to the output of f for any $(\mathsf{sk}, \mathsf{msg})$, and $\bar{g}$ accurately represents the adversary's observations.

Now we develop the notion of an **interlace**. First we will introduce a few preliminaries. Let S be the set of lists of emissions. For clarity, let $O = S$ be the set of emissions seen by the adversary. A $\bar{g}$ can be described as a function $\hat{g} : K \times M \to S$. Let $s^1, \ldots, s^m$ be the side-channel outputs of m instances of a discretized function $\hat{g}$. Let an interlace schedule $r \in R_m$ be a list, with repetition, of elements in $\{1, \ldots, m\}$.

We now describe what it means to "follow a schedule". Let $\mathcal{I} : S^m \times R \to O$ follow algorithm 1. $\mathcal{I}$ executes a schedule; in other words, $\mathcal{I}$ takes in a description of which order to run each interval for each instance (a schedule) and finds the output that would be created if the instances were run in that sequence. For example a schedule of $(1, 2, 2, 1)$ would "run" one interval of instance 1, two intervals of instance 2, then one interval of instance 1.

Algorithm 1 $\mathcal{I}(s_1, \ldots, s_m, r)$

```
for i in 1, ..., |r| do
    j = r_i
    s'_i ← pop(s_j)
end for
return s'
```

Let a scheduler be a function that creates a valid schedule; formally, a scheduler is a function $\mathcal{A}' : S^m \to I$, where each element $i \in \{1, \ldots, m\}$ appears exactly $|s^i|$ times for all $r = \mathcal{A}'(s^1, ..., s^m)$.

We define an **interlacing** as an algorithm $\mathcal{A} : S^m \to O$ is defined as:

$$\mathcal{A}(s^1, ..., s^m) = \mathcal{I}(s^1, ..., s^m, \mathcal{A}'(s^1, ..., s^m)) \tag{1}$$

We call any o output by an $\mathcal{A}$ an **interlace**.

In our model, the adversary observes only $o \in O$ output by $\mathcal{A}$. We claim that this is a reasonable assumption because we assume an observer cannot differentiate between instances without observing the outputs. This means that we assume that if there are processes P1 and P2 running on the same machine and emitting information over the same side channel, an observer cannot differentiate which process is communicating over the channel at a given time, without examining the outputs of the channel. So if P1 and P2 were both emitting "A" over the same channel, then it would be impossible to differentiate between the two instances. There are many side channels where this is true, such as the timing side channel, network side channel, and power side channel. There are side

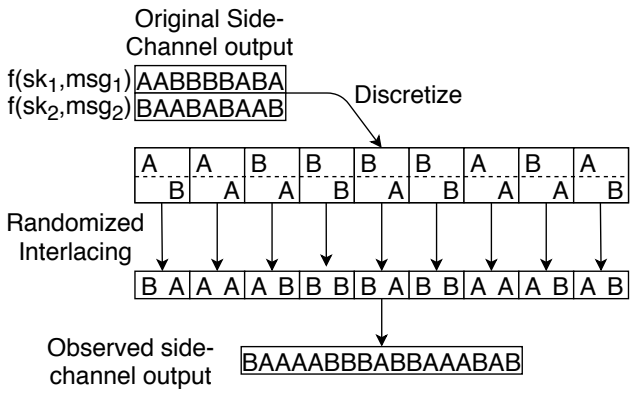

Fig. 2. The model for randomized interlacing. The two unmodified side-channel outputs are shown in the top left. These outputs are discretized. Our randomized interlaced algorithm is applied on the discretized outputs. For each timestep, all instances are executed exactly once in a random order. This results in the final, observed side-channel; that is, the adversary's videw is the bottom-most string. While we show two instances, more instances can be used for improved security.

channels where this would not necessarily be true, such as the cache side channel, where one could imagine performing a FLUSH+RELOAD attack on cache lines belonging to P1, in order to determine which process would run next.

Our model describes a Markov chain: $(K \times M)^m \rightarrow S^m \rightarrow S$. This is more concretely illustrated in figure 2. The original model describes systems of the form $K \times M \rightarrow S$, and thus by the data processing inequality[2], the information in S must be less than or equal to the information in S^m, which is the information leaked under the Köpf *et al.* model. We can quantify the loss in information in the interlace (the second transition) as some function ϵ.

Working backwards, our adversary is trying to infer an $\mathsf{sk} \in K$. In the Köpf *et al.* model, the adversary observes $s \in S$ which has not been interlaced. In our model, the adversary observes $o \in O$ which is the interlace of $s^1, \ldots, s^m$, where for any i, s^i is determined by $(\mathsf{sk}_i, \mathsf{msg}_i)$ according to Köpf *et al.*'s model. The adversary, in turn, is trying to infer sk_i for some i from o. Since s^j for $j \neq i$ is independent of s^i, recovering sk_i from o is equivalent to recovering s^i from o and recovering sk_i from s^i.

III. Randomized Interlacing

Here we introduce our primary defense for this paper. Our defense is similar to shuffling two or more decks of cards together. Our defense is equivalent to taking m "input" decks of cards, pulling the top card off each deck, shuffling those cards together, and adding those cards to an "output" deck.

In **randomized interlacing**, multiple instances of the same algorithm are executed in an alternating fashion in the same execution thread. In each step, the order of execution is randomized. The randomized interlacing algorithm is defined in Algorithm 2 and illustrated in figure 2.

[2]The data processing inequality states that for any Markov chain $X \rightarrow Y \rightarrow Z$, the following holds: $I(Y;X) \geq I(Z;X)$, where I denotes mutual information. In other words, no processing of Y can yield more information about X

Algorithm 2 $\mathsf{RandInterlacing}(\mathsf{sk}_1, \mathsf{msg}_1), \ldots, (\mathsf{sk}_m, \mathsf{msg}_m)$

1: For every i, initialize instance i with $(\mathsf{sk}_i, \mathsf{msg}_i)$
2: **while** Any instance has more steps to execute **do**
3: Choose a random permutation of $(1, \ldots, m)$
4: Execute one step of each instance in the chosen permutation
5: **end while**

A. Security

In order to establish security, we need to show that information is lost at the middle arrows, labelled "Interlace" in figure 2. We show security by proving the process of applying a randomized interlace limits the amount of information that can be transmitted over the side channel. We find that a randomized interlace limits the information transmission logarithmically with respect to the number of instances.

Formally, we assume our adversary is given an output string, o, from a randomized interlace, and their goal is to recover the input strings, $s^1, s^2, ..., s^m$ of length n. We model this transformation as a noisy channel, and compute the channel capacity. This gives an upper bound on the information, in terms of Shannon entropy, the adversary can learn. In all cases, $\log$ is in base 2.

Our randomized interlacing technique has the following additional assumptions about the structure of the side-channel:

1) Let S_f be the set of emission sequences from f. The most important assumption is that $\exists c \; \forall s \in S_f, |s| = c$; that is, all emissions sequences are of constant length.
2) The set of possible emissions for each interval should be small. More formally, let $S_{f,i} = \{s_i | s \in S_f\}$. We call $S_{f,i}$ the "alphabet" of the emission. We say $a = \max_i |S_{f,i}|$ is the "alphabet size". Theorems 1, 2 require $a = 2$. Conjecture 3 generalizes Theorem 2 to $a \in \mathbb{Z}^+$. This is important to minimize the Shannon entropy of each emission. Note that different intervals need not share the same alphabet.

We provide two results based on slightly different assumptions. We provide intuitions for the results, and leave the proofs to the appendix. Theorem 1 demonstrates bounds without assuming independence of the inputs; Theorem 2 assumes the inputs are independent, giving more favorable bounds. We generalize Theorem 2 to arbitrary-sized alphabets in conjecture 3.

Figure 3 shows how the leakage per input bit changes as we increase m. As we can see, diminishing returns quickly set in.

Theorem 1: Given a randomized interlace o as defined in algorithm 2, no adversary can recover more than ϵ-bits of information from the strings $s^1, \ldots s^m$, where

$$\epsilon = n \cdot \log(m+1) \tag{2}$$

where $H_B(\cdot,\cdot)$ is the entropy of a binomial distribution and where $(s^1, \ldots, s^m)$ is drawn from S^m.

We show that over an m length window which is analogous to a discrete memoryless channel, we can partition the interlace

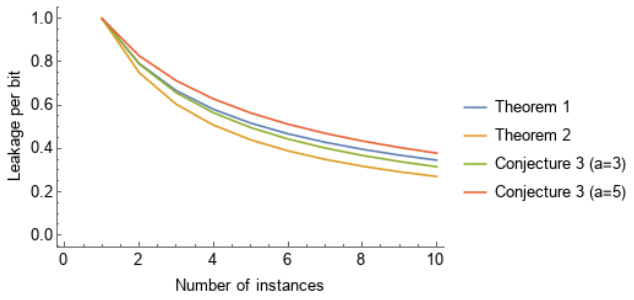

Fig. 3. The maximum information leakage per bit of input versus the number of instances. The blue line shows the leakage per bit under the assumptions from Theorem 1, and the orange from Theorem 2. The green and blue lines show the leakage per bit output from conjecture 3, under the assumption of alphabets of size 3 and 5, respectively. In each case, the leakage per input bit declines sharply, but diminishing returns set in quickly.

into $m+1$ sets of outputs, such that in each partition each output is equally likely. Then we argue that the most efficient way to transmit information over such a channel is to have uniform probability of emitting each symbol at each interval. One such approach is one input per partition, giving $m+1$ inputs that each map to their own distinct partition, for $\log(m+1)$ bits.

Theorem 2: Given a randomized interlace o as defined in algorithm 2, no adversary can recover more than ϵ-bits of information from the strings $s^1, \ldots s^m$, where

$$\epsilon = n \cdot H_B(m, \frac{1}{2}) \tag{3}$$

where $H_B(\cdot,\cdot)$ is the entropy of a binomial distribution and where each s^i are drawn from S independently.

Our proof again involves computing the channel capacity. A better intuition for the theorem statement is that an adversary who learns the original strings also learns information about the ordering, so the channel capacity must be reduced to account for this additional information. To find this reduction, we show that each bit of each string can be described as a random variable with a constant probability, and we show that no such probability distribution allows for a channel capacity exceeding $\frac{1}{2^m}\sum_{k=0}^{m}\binom{m}{k}\left(m-\log\binom{m}{k}\right)$ bits.

1) Generalizing: In the case where the alphabet of emissions is larger than 2, our model does not describe the leakage.

Conjecture 3: Let all emissions be drawn from an alphabet of size a. Given a randomized interlace o as defined in algorithm 2, no adversary can recover more than information from strings $s^1, \ldots, s^m$, where

$$\epsilon = H_M\left(m, \left(\frac{1}{a}, \ldots, \frac{1}{a}\right)\right) \tag{4}$$

where $H_M(m, \boldsymbol{p})$ is the entropy of the multinomial distribution, and where each s^i is drawn from S independently.

We believe this conjecture to be true because this is the channel capacity if uniform draws maximize the mutual information between S and O. This was true for the $a=2$ case, so it may be true for the multinomial case.

B. Algorithm correctness

It is worth considering how our technique affects the correctness of an algorithm. It is necessary that each instance be sandboxed in such a way that instances do not have side-effects and interfere with other instances' execution. The extent of the sandboxing can be minimized with careful software design, with a pure function requiring no sandboxing. In the case where some of the data observed by the adversary is also needed for the algorithm to function, for example in the case of video fingerprinting where the adversary observes the communication between the sender and receiver, the receiver must know the schedule so they can disentangle the decoy instances from the real instances. In the case where the the algorithm does not interact with the data observed by the adversary, for example the power side channel of a cryptography algorithm, the correctness of the algorithm is ensured with sandboxing alone.

IV. Inverse Interlacing

Here we introduce another interlacing we call **inverse interlacing**. Inverse interlacing provides complete information-theoretic security against the targeted side channel attacks, but requires strict conditions to be met.

We can describe "constant X" defenses – constant time, constant size, constant power, etc. – using our model by making some additional assumptions about the function. In particular, we assume that for any input pair sk, x we can efficiently identify an input pair sk', x' such that for any emission for the first input pair is the complement of the corresponding emission for the second input pair. This can be extended to larger emission spaces by increasing the number of pairs and choosing additional sk', x' pairs, such that at every emission step, every element in the corresponding emission space is present exactly once. The interlacing technique is then applied.[3]

More formally, inverse interlacing requires the following assumptions to be true.

1) Let S_f be the set of emission sequences from f. The most important assumption is that $\exists c\ \forall s \in S_f, |s| = c$; that is, all emissions sequences are of constant length.
2) We assume we can always find an "inverse" $(\mathsf{sk}', \mathsf{msg}')$ pair to a given $(\mathsf{sk}, \mathsf{msg})$; that is, inputs that output the opposite value at each interval (or complete the full set, for $a > 2$). More formally, we assume there exists some function $\mathsf{inv}_f : K \times M \to (K \times M)^{a-1}$ such that $\forall i \in \{1, \ldots, m\}$

$$f(\mathsf{sk}, \mathsf{msg}))_i \cup \{f(\mathsf{inv}_f(\mathsf{sk}, \mathsf{msg})_1)_i, \ldots, f(\mathsf{inv}_f(\mathsf{sk}, \mathsf{msg})_{a-1})_i\} = S_{f,i} \tag{5}$$

Note that this assumption implies $m = a$.

Inverse interlacing is defined by the following algorithm:

We also present the following, formal bounds on inverse interlacing:

[3]Note that, here, the scheduler need not be random. For example, a sorting based on emitted value is an acceptable output schedule. For simplicity, we choose to use a random schedule.

Algorithm 3 $\mathsf{InvInterlace}(\mathsf{sk}_1, \mathsf{msg}_1)$

1: $(\mathsf{sk}_2, \mathsf{msg}_2), \ldots, (\mathsf{sk}_m, \mathsf{msg}_m) \leftarrow \mathsf{inv}(\mathsf{sk}_1, \mathsf{msg}_1)$
2: **return** $\mathrm{RandInterlace}((\mathsf{sk}_1, \mathsf{msg}_1), \ldots, (\mathsf{sk}_m, \mathsf{msg}_m))$

Theorem 4: Given an inverse interlace o as defined in algorithm 3, no adversary can recover more than ϵ-bits of information from the strings s, where

$$\epsilon = 0 \tag{6}$$

Proof: Consider any window, as defined in the proof of Theorem 1. The adversary's view is always identical: a random ordering of each emission in the alphabet, exactly once. As we discussed previously, these observations are identical from the adversary's perspective. Shannon entropy is defined as $H(P) = -E_P[\log P]$, and since there is only one possible observation, the output has 0 entropy.

V. Case Study: Video Traffic Fingerprinting

In the section, we explore how our defense can be used to tackle the problem of traffic analysis of video streaming. In order to approach the problem, we focus on the attack presented by Schuster *et al.* [13]. We summarize the attack in section VI. To model the leakage, we need to model the side channel under our model. Variable bitrate is ultimately encoded by dividing the video into segments of constant time duration, but variable data size. The client requests the a segment when their current time marker advances sufficiently. In our simulated case study, we bucket and mitigate the bitrate observations directly. In a real-world setting, bitrate is not as easy to bucket, but we can bucket and mitigate a video segment size channel, thereby indirectly bucketing and mitigating the bitrate channel. This model naturally lends itself to discretization, by defining the transmission of one segment as an discretization interval.

To limit the alphabet size, the video segments sizes can be bucketed, using a technique similar to Köpf *et al.*, 2009 [8] to reduce the output alphabet for each interval. The size of the alphabet provides a trade-off between performance and security. Conjecture 3, if correct, can be used to determine the impact of larger alphabets on security. In a real-world setting, videos should also be bucketed by total duration so-as not to leak video duration.

Note that under our model, it is always possible to obtain perfect true negative rate – that is, an adversary can always rule out if you watched a video. However, as long as the set the video is drawn from is sufficiently large relative to the leakage, any classifier would have a high false positive rate.

As we've mentioned, side channel attacks that attempt to determine which video is currently being streamed typically work by attempting to exploit variable bitrates. For our simulation, we'll break up the stream into intervals of fixed length, and then bucket the bitrates.

In figure 4, we can see an example of a video with a variable bitrate, before and after rounding up to our fixed alphabet. We selected an alphabet of {100000, 175000, 275000} Bits/second, and rounded up the bitrate of each interval of one second, to the next term in the alphabet. Next, we took

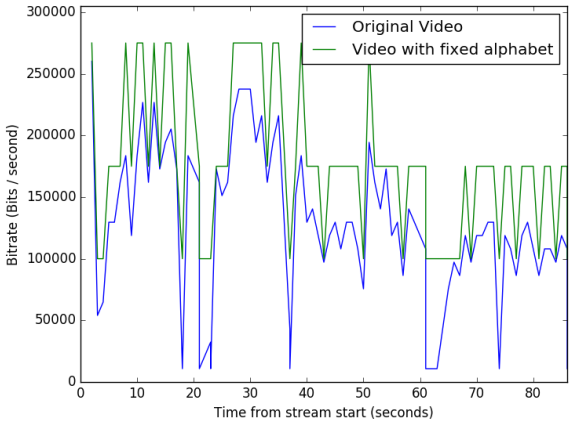

Fig. 4. An example of the bitrate of an arbitrarily selected YouTube video, before and after rounding up the bitrate

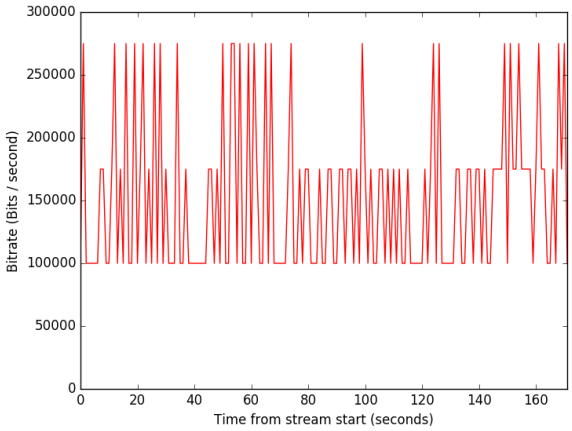

Fig. 5. An example of our complete interlace of two selected YouTube videos

a second video, performed the same operation with the same alphabet. We then follow our first algorithm, by computing an interlace schedule. This is just a string of randomly generated bits equal to the number of intervals in both videos. We then iterate through the schedule and both videos, constructing a new streaming schedule. We follow our algorithm by appending the corresponding intervals of each video in the order determined by our schedule, constructing a final interlace, that is visible in figure 5.

While we can see the traffic of our original video in figure 5, we also see that a significant amount of decoy data has also been injected. In the high bitrate section towards the beginning, we see the presence of both low and high bitrates. Towards the end, where the bitrate is lower, we see new spikes. The introduced highs and lows along with the uncertainty of which observation belongs to which video will lead to false positives. The expected number of false positives will depend on the set of possible videos, their likelihoods, the number of decoy instances, the chosen alphabet, and the number of emissions. As with all side-channel attacks, we are looking at a perfect view—a real world adversary will experience a noisy view.

VI. Related Work

To establish provability in any field, an accurate, formal model of the questions asked is needed. For side-channel attacks, one such formal model of side-channel attacks was introduced by Köpf *et al.*, 2007 [7]. Their model describes side-channel attacks both adaptive and non-adaptive adversarial strategies. Köpf *et al.*, 2009 [8] demonstrates the power of such methods by introducing a straightforward defense mechanism and showing how a defender can configure a performance-security trade-off using their model.

Heuristic defenses face an increasing threat as machine learning becomes more capable of detecting weaknesses. Sirinam *et al.* [16] presented a deep learning approach to website fingerprinting against the Tor network. Under some conditions, they were able to obtain an accuracy approaching 100%. Askarov *et al.* [3] introduced a class of blackbox, composable mitigators against timing channels. Askarov *et al.*'s analysis also uses information theory to provide provable bounds on side channel leakage.

The increasing awareness of side channel attacks has lead to the introduction of defenses that minimize the work on the cryptographer and the developer. Shelton *et al.* [15] presented a code rewrite engine that automatically simulates the power side-channel and rewrites segments that appear to leak information.

Video fingerprinting is an example of a side channel attack that is very difficult to mitigate in practice without severe performance penalties. Recent work [12] has shown that it is possible to determine which specific video on Netflix is being streamed, even through an encrypted connection. This poses significant privacy implications for video streaming sites going forward, which are nontrivial to solve. In many previous side channel defenses for other domains, constant time operations are enforced, resulting in effectively zero information flow. This is used in libraries such as OpenSSL [1]. These techniques don't generalize well to defenses for video fingerprinting, as the performance overhead of having a constant streaming bitrate is too large.

Past work such as Schuster *et al.* [13] described a side channel attack against encrypted video streams. In their attack, the adversary watches the changes in bitrate and compares the pattern against an established database of video bitrate patterns. The video is then assumed to be the video in the database with the most similar bitrate pattern. This is effective because modern video streaming divides videos into constant time segments, and only transmits the segment when the segment is temporally close to the current video time.

The most obvious defense to such an attack is to make bitrate constant. This can be accomplished by fixing the bitrate to a set value [11], or by injecting random quantities of noise to hide bitrate [17]. A method known as dependent link padding was introduced to compromise between the privacy of constant bitrates with the change in bitrate demand of real networks [19].

There are several defenses that could be applicable to this and similar attacks. One such defense was proposed by Zhang *et al.*, 2013 [20] and protects encrypted channels against side-channel attacks by multiplexing multiple channels through a single one. Another related defense was proposed by Wang *et al.* [18] which injects enough fictitious data to provide a believable "cover story" for any access. While their technique is targeted towards static content, it could be adapted for video streaming. A more recent paper, Zhang *et al.*, 2019 [21] targets video streaming directly. They provide two mechanisms for defending against attacks. The first is to inject enough traffic such that the input the attacker finds is an adversarial example to their machine learning algorithm. The second defense is to introduce differential privacy into streaming bitrate, such that videos with similar bitrate profiles are statistically indistinguishable.

VII. Limitations

Our defense has limitations that need to be carefully considered before implementation. First and foremost, the system must, by design, have an overhead of 2x or more, scaling linearly with the number of decoy instances. This can be mitigated if the alternate instances can be put to use (e.g., if multiple videos are being streamed simultaneously), but that's not always possible.

There are some cases where the adversary can obtain enough entropy to discover the key, from an information-theoretic perspective. If the entropy of the secret is low relative to the size of the output observed, for example if the key has 128 bits and the adversary observes 300 bits per instance from the side channel, then the adversary may still gather enough entropy to recover the secret. Furthermore, if the adversary is allowed multiple observations of the same secret's interlaced side channel output with different public inputs or different cover instances, then the adversary may be able to gather enough entropy to learn the secret. This is not the case if the side channel output is independent of the message and the cover instances chosen depend on the secret alone.

We leave the computational hardness of disentangling interlaces an open problem. If it is "easy" to disentangle interlaces given sufficient information, then this would leak the original side-channel output, allowing the adversary to recover the scheme using a traditional side-channel attack. For this reason, it is important to take care when using the same secret multiple times.

Finally, our algorithm cannot defend against an adversary that manages to find an unanticipated source of information, for example an undefended, alternate channel or a more fine-grained observation of the targeted channel. Additionally, the scheduling algorithm and the context switcher might be vulnerable to side channel attacks, so care needs to be taken when implementing this system to avoid directly leaking information about which instance is running.

VIII. Future Work

In a followup work, we intend to propose a related technique we call "single shuffling", where the lock-step restriction is removed from the randomized interlacing. Crucially, this requires assuming all intervals share an alphabet. Additionally, it is an open problem to compute information-theoretic bounds on the leakage, although we expect them to be significantly better. We additionally intend to perform a full study of

how effective our methods are in practice—both in terms of information leaked and performance.

An open question is how the security of our techniques changes over repeated rounds. It seems feasible that even if the full information of the key is leaked over multiple rounds, there may be additional computational hardness protections. Furthermore, it has yet to be determined how to compose the information leaked over multiple, non-identical rounds. It seems likely that there may be mutual information leaked between the rounds. An information theoretic framework for quantifying the mutual leakage between rounds would be important for many cryptographic applications.

Another open question is how our defense scales in terms of efficiency to other side channel defenses. We expect our "single shuffling" technique will greatly decrease the number of instances required. However, at a minimum, we will always need two instances. If we can develop a system that always interlaces real instances, rather than creating random, cover instances, we can further reduce the cost of our method.

ACKNOWLEDGMENT

REFERENCES

[1] OpenSSL Project. [Online]. Available: https://openssl.org

[2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 29–45.

[3] A. Askarov, D. Zhang, and A. C. Myers, "Predictive black-box mitigation of timing channels," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 297–307.

[4] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701–716, 2005.

[5] B. den Boer, K. Lemke, and G. Wicke, "A dpa attack against the modular reduction within a crt implementation of rsa," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 228–243.

[6] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*. Springer, 1999, pp. 388–397.

[7] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 286–296.

[8] B. Köpf and M. Dürmuth, "A provably secure and efficient countermeasure against timing attacks," in *2009 22nd IEEE Computer Security Foundations Symposium*. IEEE, 2009, pp. 324–335.

[9] P. Mateev, "On the entropy of the multinomial distribution," *Theory of Probability & Its Applications*, vol. 23, no. 1, pp. 188–190, 1978.

[10] D. Page, "Theoretical use of cache memory as a cryptanalytic side-channel." *IACR Cryptology ePrint Archive*, vol. 2002, no. 169, 2002.

[11] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Isdn-m ix es: Untraceable communication with very small bandwidth overhead," in *Kommunikation in verteilten Systemen*. Springer, 1991, pp. 451–463.

[12] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate dash videos streamed over encrypted 802.11 n connections," in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 1107–1112.

[13] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1357–1374.

[14] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[15] M. A. Shelton, N. Samwel, L. Batina, F. Regazzoni, M. Wagner, and Y. Yarom, "Rosita: Towards automatic elimination of power-analysis leakage in ciphers," Cryptology ePrint Archive, Report 2019/1445, 2019, https://eprint.iacr.org/2019/1445.

[16] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 1928–1943.

[17] P. Venkitasubramaniam, T. He, and L. Tong, "Relay secrecy in wireless networks with eavesdroppers," in *Proceedings of the Allerton Conference on Communication, Control and Computing*, 2006.

[18] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1375–1390.

[19] W. Wang, M. Motani, and V. Srinivasan, "Dependent link padding algorithms for low latency anonymity systems," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 323–332.

[20] F. Zhang, W. He, Y. Chen, Z. Li, X. Wang, S. Chen, and X. Liu, "Thwarting wi-fi side-channel analysis through traffic demultiplexing," *IEEE Transactions on Wireless Communications*, vol. 13, no. 1, pp. 86–98, 2013.

[21] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang, "Statistical privacy for streaming traffic." in *NDSS*, 2019.

APPENDIX

A. Notation guide

Symbol	Meaning
m	The number of instances for an interlace
n	The length of the interlace
a	The size of the emission alphabet
sk	A secret input
msg	A public input
s	The output of a side-channel
o	The output of an interlace
e	An emission, or output of a single interval of a discretized side channel
st	A program state
K	The space of secret inputs
M	The space of public inputs
S	The space of side-channel outputs
O	The space of interlace outputs
E	The space of emissions
ST	The space of program states
f	The side-channel of an algorithm, under the Köpf *et al.* model
$\bar{g}$	A discretized side-channel. Specifically, a list of emission functions
g_i	The emission function at interval i for a discretized side channel
$\mathcal{A}$	An interlace
$\mathcal{A}'$	A scheduler for an interlace
$\mathcal{I}$	The schedule executor

B. Proof of theorem 1

Consider the observed side channel output o from a randomized interlace. Choose some $i \in (1, \ldots, n)$. Let $w_i = \left(o_{m(i-1)+1}, ..., o_{mi}\right)$. Observe that w_i depends only on $b_i, s_i^1, s_i^2, ..., s_i^m$. We can model the relationship between w_i and $s_i^1, s_i^2, ..., s_i^m$ as a discrete memoryless channel. Specifically, such a channel is defined by an input alphabet $\mathcal{X}$, an output alphabet $\mathcal{Y}$, and a transition probability matrix $p(y|x)$ for $x \in \mathcal{X}, y \in \mathcal{Y}$.

Let $\mathcal{X} = \mathcal{Y} = \{0,1\}^m$. Given some x, let $k = \mathsf{count}(1, x)$. Then

$$p(y|x) = \begin{cases} \binom{m}{k}^{-1} & \mathsf{count}(1,y) = \mathsf{count}(1,x) \\ 0 & \text{otherwise} \end{cases}$$

We can see that the channel models the interleave since x can only transition to y if they have the same number of 1s, the probability of transition to each y' with the same number of 1s is uniform, and there are $\binom{m}{k}$ ways to arrange k 1s across m slots.

To find the maximum information the adversary can recover from one window, we need only find the maximum information transmittable over the channel, which is it's channel capacity. This is defined by:

$$C = \max_{p(\mathcal{X})} I(\mathcal{X};\mathcal{Y})$$

We can express this as:

$$C = \max_{p(\mathcal{X})} \sum_{x,y} p(x)p(y|x) \log \frac{p(y|x)}{p(y)}$$

We can partition this sum into different values for $\mathsf{count}(1,x)$. Note that if $\mathsf{count}(1,x) \neq \mathsf{count}(1,y)$, then $p(y|x) = 0$, and hence so the corresponding term is 0. Hence, the channel capacity expression reduces to:

$$C = \max_{p(\mathcal{X})} \sum_{k=0}^{m} \sum_{\substack{x,y \\ \mathsf{count}(1,x)=k \\ \mathsf{count}(1,y)=k}} p(x)p(y|x) \log \frac{p(y|x)}{p(y)}$$

We can see that $p(y) = \sum_x p(x,y) = \sum_{\substack{x,y \\ \mathsf{count}(1,x)=k}} p(x)\binom{m}{k}^{-1}$, where $k = \mathsf{count}(1,y)$, since the probability of finding an x that could transition to y is $\sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x)$ and the probability of transitioning is $\binom{m}{k}^{-1}$. Hence,

$$C = \max_{p(\mathcal{X})} \sum_{k=0}^{m} \sum_{\substack{x,y \\ \mathsf{count}(1,x)=k}} p(x) \binom{m}{k}^{-1} \log \frac{\binom{m}{k}^{-1}}{\sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x)\binom{m}{k}^{-1}}$$

$$= \max_{p(\mathcal{X})} \sum_{k=0}^{m} \binom{m}{k} \binom{m}{k}^{-1} \sum_{\substack{x \\ \mathsf{count}(1,x)=k}} -p(x) \log \sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x)$$

$$= \max_{p(\mathcal{X})} -\sum_{k=0}^{m} \log \left(\sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x) \right) \sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x)$$

Let Z be a random variable that draws with some probability from the set $\{z_0, z_1, ...z_m\}$, such that

$$p(z_k) = \sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x)$$

Because of our partitioning scheme, $\sum_{k=0}^{m} p(z_k) = 1$ and $0 \leq p(z_k) \leq 0$, Z constitutes a random variable. Then, with a change of variables, we have:

$$C = \max_{p(\mathcal{X})} \sum_{k=0}^{m} p(z_k) \log p(z_k)$$
$$= \max_{p(\mathcal{X})} H(Z)$$

Where $H(Z)$ is the Shannon entropy of the random variable Z. We know from information theory that this value is maximized when Z is uniformly distributed. Since $p(Z)$ depends on $p(X)$ and $H(Z)$ does not depend on X aside from in that it depends on Z, we can say that

$$\max_{p(\mathcal{X})} H(Z) = \max_{p(Z)} H(Z)$$
$$\leq \log(m+1)$$

Where the final inequality follows from the fact that the support of Z is over $m+1$ points.

We then need to apply this to each possible value of i. We sum the values for each window, giving us a total channel capacity for the entire string lengths of $n \cdot \log(m+1)$.

C. Proof of theorem 2

Proof: Because each string is drawn independently, then if we examine the set of all possible strings, S, then for some position j, we can determine the probability that position has the value 1, which we will denote q. This can be done by finding the fraction of strings that have 1 at position j. It then holds that for that position, the probability of seeing a 0 is $1-q$. Because each string is drawn independently, the values s_j^i are mutually independent and are each 1 with probability q.

This observation allows us to place a restriction on the distribution of probabilities when computing the channel capacity: $p(x) = q^{\mathsf{count}(1,x)}(1-q)^{|x|-\mathsf{count}(1,x)}, 0 \leq q \leq 1$. Then, maximizing C with respect to $p(\mathcal{X})$ is the same as maximizing C with respect to q.

$$C = \max_{p(\mathcal{X})} \sum_{x,y} p(x)p(y|x) \log \frac{p(y|x)}{p(y)}$$
$$= \max_{q} \sum_{x,y} p(x)p(y|x) \log \frac{p(y|x)}{p(y)}$$

We can use some of the work from the proof of Theorem 1 to obtain the following expression for the capacity after substituting $p(x) = q^k(1-q)^{m-k}$:

$$C = \max_{q} -\sum_{k=0}^{m} \log \left(\sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x) \right) \sum_{\substack{x \\ \mathsf{count}(1,x)=k}} p(x)$$

$$= \max_{q} -\sum_{k=0}^{m} \log \left(\sum_{\substack{x \\ \mathsf{count}(1,x)=k}} q^k(1-q)^{m-k} \right) \sum_{\substack{x \\ \mathsf{count}(1,x)=k}} q^k(1-q)^{m-k}$$

$$= \max_{q} -\sum_{k=0}^{m} \log \left(\binom{m}{k} q^k(1-q)^{m-k} \right) \binom{m}{k} q^k(1-q)^{m-k}$$

Where the final expression is simply the entropy of the binomial distribution, $H_B(m,q)$, which is known from prior

work [9] to be maximal at $q = \frac{1}{2}$. As before, summing over all n positions, we obtain $C = n \cdot H_B(m, \frac{1}{2})$ which completes the proof.