Probabilistic Computation for Information Security

Piotr Mardziel
Department of Computer Science
University of Maryland, College Park
College Park, MD 20740
piotrm@cs.umd.edu

Kasturi Rangan Raghavan
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024
kr@cs.ucla.edu

Probabilistic computation is a convenient means of mechanically reasoning about a variety of information security problems. At its core, information security concerns itself with measuring or limiting the knowledge an adversary might attain from interacting with a protected system. Probabilistic inference lets one compute this knowledge explicitly as long as the interaction can be described as a program in a suitable probabilistic language that supports conditioning. Security concerns, however, require soundness guarantees on probabilistic inference which are generally not present in machine learning applications. We summarize some recent work on probabilistic computing for information security and highlight challenging aspects that still need to be addressed.

Tracking knowledge Tracking adversary knowledge is a central tool for the applications to follow. We describe this core concept adopting the notation and conventions of Clarkson's experimental protocol [2].

Alice wishes to grant Bob partial access to some secret data fields while making sure Bob does not learn too much. The data here is secret in that Bob does not know it, though it is not necessarily data that might be considered secret in the typical sense of the word PM – Added this but kept the word "secret" to not confuse people in the discussions that follow.. To do this, Alice will track what Bob learns about these fields as a rational adversary who will use any information provided to him in order to learn Alice's secret values. Specifying the protection of the secret will be addressed shortly.

Alice's secret data is a value s^* which Bob knows is in some set of possibilities **Secrets**. Furthermore, Bob considers some values more likely than others, a probability distribution $\delta_1 : \textbf{Secrets} \rightarrow \mathbb{R}$. We will refer to this distribution as Bob's *belief*. Alice then considers the effect of letting Bob learn the output of a potentially non-deterministic function W, written in a probabilistic language. She can do this by computing the conditional distribution $\delta_2 \overset{\text{def}}{=} \delta_1|(W(s^*) = o)$ for some potential output value o. Assuming $\delta|C$ indeed implements *exact* probabilistic conditioning, this distribution δ_2 will be the belief Bob would attain, were he to see that the function W outputs o given input s^*.

If Bob indeed does learn this output, Alice can revise her representation of Bob's belief to δ_2 which she can then revise further if Bob learns additional functions over her data. In this manner Alice explicitly tracks what Bob knows about her secret as a probability distribution. The problems of how Alice knew δ_1, Bob's initial belief, in the first place and whether she can actually perform exact inference will need to be addressed. PM – Remove or add discussion about initial belief.

Knowledge-based policy enforcement Alice can use the tracked belief in a variety of ways, the most direct of which is to judge the information security or safety of Q in order to determine whether she should let Bob get access to the output of the function. To do this, Alice starts by defining a knowledge-based policy $P : \textbf{Dist} \rightarrow \{\mathsf{True}, \mathsf{False}\}$, where **Dist** is a set of all distributions over **Secrets**. For example, Alice might require that Bob never becomes too certain of her value, so she defines $P(\delta) \overset{\text{def}}{=} \forall s \in \textbf{Secrets} \,.\, \delta(s) \leq t$ for some threshold t. She will then deem the query Q safe whenever $P(\delta_2)$ holds. This particular policy is an application of Smith's vulnerability metric [6], which measures the expected likelihood of Bob guessing a secret value correctly in one try.

Unfortunately for Alice, exact probabilistic conditioning is expensive so she decides has to use an approximation. Instead of computing δ_2, she gets $\delta_2^{\sim} \overset{\text{def}}{=} \delta_1|^{\sim}(W(s^*) = o) \approx \delta_1|(W(s^*) = o)$.

The approximation $\delta_2^{\sim}$ here need not be a proper probability distribution. She does not want the approximate inference to conclude a query is safe when it really is not so she requires the approximation to be sound relative to her policy:

Definition. A probabilistic inference method $|^{\sim}$ is *sound relative to a policy* P iff for every belief δ, if $P(\delta|C)$ fails then $P(\delta|^{\sim}C)$ fails.

For Alice's example policy, a sound inference method would have to never underestimate the probability of any secret, though it could over-approximate it. In recent work [5], we demonstrate a probabilistic language capable of this approximate, but sound inference. Given the definition of conditional probability, $Pr(A|B) = Pr(A \wedge B)/Pr(B)$, the probabilistic interpreter maintains an over-approximation of $Pr(A \wedge B)$ and an under-approximation of $Pr(B)$, leading to a sound upper bound for $Pr(A|B)$. The work shows how Alice can enforce her policy without revealing anything about her secret to Bob. Further application of these ideas were proposed to protect information of multiple mutually distrusting parties participating in secure multi-party computation [4].

If Alice deems W safe, she can allow Bob access to $W(s^*)$ and update her representation of Bob's belief to $\delta_2^{\sim}$. If she rejects W, she will not allow access to $W(s^*)$ and keep δ_1 as the representation of Bob's unchanged belief. Whether Alice deems W safe or not, she can repeat the same process for another program that might come around, starting from either δ_1 or $\delta_2^{\sim}$, thereby maintaining an explicit representation of Bob's knowledge while enforcing her knowledge-based policy.

Obfuscation/Noising Probabilistic computation is also very convenient for discerning the effects of obfuscation or noising mechanisms, via simple composition with the observation function W.

If Alice does not want Bob to learn the output of W directly, she can add obfuscation to its input, or noising to its output. Let O be a potentially non-deterministic obfuscation function, written in a probabilistic language, that takes in a value $s \in$ **Secrets** and produces another $s' \in$ **Secrets**. Depending on the structure of **Secrets**, this obfuscation might be in the form of resolution reduction, permutation of locations, or a variety of other mechanisms. Alice can then consider Bob's knowledge were he to learn the output of an obfuscated function $W(O(s^*))$ instead of $Q(s^*)$ directly, using conditioning, $\delta_2 = \delta_1 | \left(W(O(s^*)) = o\right)$.

Alice can also consider adding noise to the output of W, using a noising function N, again written in a probabilistic language. She can then consider Bob's revised belief $\delta_2 = \delta_1 | \left(N(W(s^*)) = o\right)$. She might consider obfuscation and noise at the same time, revising Bob's belief to $\delta_2 = \delta_1 | \left(N(W(O(s^*))) = o\right)$. Bob's presumed revised belief assumes that he knows O and N.

Blacklist/Whitelist functions Having a means of determining the effects of obfuscation and noising allows one to design such functions to satisfy security objectives. It is easy to create a trivial noising function, $N(s) \overset{\text{def}}{=} 42$, if only security is an issue, thereby ensuring Bob learns absolutely nothing. This ignores utility concerns that Bob (and Alice) might be interested in.

Though still cautiously considered an adversary intent on learning s^*, let us assume that Bob's utility is derived from the accuracy or precision of his view on $W(s^*)$, and not s^*. Depending on the structure of W, it is certainly possible to reveal $W(s^*)$ without revealing much about s^*. Likewise, Alice might not consider her hidden value s^* itself all that damaging, but could define a risky function B whose output she would like to protect. Specifying Alice's risk in terms of functions is a more versatile approach than the policies mentioned earlier that limit Bob's view on Alice's secret itself, assuming she has specific risks in mind.

The pair B and W, referred to as "blacklist" and "whitelist" functions respectively, thus specify the Bob's utility and Alice's risk in information sharing. In the framework of belief tracking via probabilistic computation, Alice can judge the safety and utility inherent in Bob's belief δ about her secret by evaluating the blacklist and whitelist functions on the probabilistic value that represents Bob's belief, written $B(\delta)$, a distribution on the outputs of B, and $W(\delta)$, a distribution on the outputs of W, respectively.

Definition. Blacklist/whitelist functions B and W are satisfied on belief δ relative to security policy P and utility policy U whenever $P(B(\delta))$ and $U(W(\delta))$ hold.

Alice can determine the safety of $B(\delta)$ and utility of $W(\delta)$ in a variety of ways. She could use the policy P from earlier on δ_B, bounding Bob's chances of guessing the output of B. She can also use the negation of P, specifically $U(\delta) \overset{\text{def}}{=} \neg P(\delta) = \exists o \;.\; \delta(o) > t$, to measure some minimum level of utility of the whitelist function W, by checking that $U(\delta_W)$ holds. Alternate means could involve the ubiquitous entropy quantity to measure a lower bound on the entropy of δ_B and an upper bound on the entropy of δ_W.

Having the ability to measure the secrecy and utility of Bob's belief, Alice can then determine how to construct a function $Q(s) = N(W(O(s)))$ to ensure $\delta| \, (Q(s^*) = o)$ satisfies the security and utility policies (if possible).

PM – Move or reword this para. Or perhaps it is no longer necessary given revised discussion above. Consider the situation where Alice possesses some data d, and Bob would like to evaluate the whitelist function W that derives Bob's utility, $o = W(d)$. Here Alice's data is not secret, instead we define a blacklist function B that serves to extract Alice's secret information, i.e. $s \overset{\text{def}}{=} B(d)$. Alice tracks Bob's belief δ_d, and Alice would like to ensure before sharing the output $W(d)$ that $\delta_d|^{\sim}(W(d) = o)$ satisfies policy $P(B(\delta_d))$.

PM – todo: There's an issue of accuracy of belief vs. uncertainty of belief. Let us ignore this issue.

PM – The below is kind of vague and I'm wondering how to (1) explain it better, and (2) fit it in better with what has just been said.

Consider that obfuscation and noising functions are specified as probabilistic programs with free parameters. Then we ask whether probabilistic computation can be leveraged to *infer* a choice of free parameters which satisfy the security and utility policies. In this way, we propose probabilistic computation as a suitable platform not only to describe and enforce security policies, but also as a mechanical and principled way to learn noising and obfuscation functions.

Predictive models for blacklist functions The blacklist (and whitelist) functions are useful to protect information which is not part of the secret value s^*, but is somehow correlated with the secret value. For example, Alice's simple demographical information like age, postal code, and gender might compose the structure of s^* but Alice might be more interested in protecting her religions preference, even though this is not a field in the system she is securing.

Predictive models or classifiers, however, might exist that do a reasonable job of predicting religion given demographical information. Naturally, Alice could specify one such model as her blacklist function B. Unfortunately this does not stop Bob from using a different model B which is as good or better at predicting Alice's religion. This is especially problematic when there are two disjoint sets of fields in Alice's secret s^* that are both good for predicting religion. Alice might obfuscate one, but leave the second set completely untouched.

In general it is impossible for Alice to ensure *all* possible models for religion are defeated by her noising/obfuscation functions. This is simply due to Bob's potential model $B(s) = r$ where r is a constant that just happens to be Alice's religion. To make this problem reasonable, we restrict ourselves to models that Bob, assumed rational, could produce, given some public body of data available to both Alice and Bob.

Background knowledge and future work An assumption on the common background data is useful for both limiting the sorts of blacklist functions B Alice should be concerned about and for determining what Bob's initial belief δ about Alice's secret fields can be.

If we assume Bob takes advantage of a common labeled dataset D_r of tuples (s_i, r_i) tying demographical information to religion, Bob can learn a variety of predictive models for religion via supervised machine learning techniques. Alice would like to therefore protect herself (via blacklist) against any model that from Bob's point of view is accurate (in that it does well on predicting religion in D_r). How to do this effectively is a challenging aspect we would like to address in the future.

A related problem is how Alice can construct δ, the initial belief Bob has about her secret value. The use of probability distributions to model Bob's knowledge already presumes that Alice's secret value s^* is a sample from δ. It is natural, then, to presume a shared data set D instances s_i among the population Alice belongs to, is also composed of samples from that distribution.

Alice could thus assume Bob would form an initial belief about her via a statistical model with some unknown parameters p with an initial distribution on p, labeled δ_p and sampling function S, written in a probabilistic language, that produces samples $s \in$ **Secrets** given parameters p. Alice could then infer these parameters by conditioning on the samples from this distribution being D, specifically $\delta_p|(S(p) = s_1)|(S(p) = s_2)| \cdots$ as is demonstrated in recent work of Gordon et al.[3]. Bob's initial belief about s^* would then be the distribution $S(\delta_p)$.

The problem here is the same as in the problem of specifying blacklist functions; Alice does not know what model Bob could use, either to infer her religion, or to form an initial belief. Alice could take into account multiple models, thereby having to track several options for Bob's belief, but how to construct a sufficient set of models is another challenging element of our future work.

PM – todo: Need to bring out the point that for having a reasonable definition of security and utility, it is beneficial to include a *diverse* set of blacklist and whitelist functions.

PM – Can we get away from having to talk about a set of functions here, I don't think they are strictly necessary. Also, for the reference, was the work specific to a domain (of sensor readings or some such) or more general that applies to a broad type of data? Can you mention this?

From a set of blacklist functions **Blacklists** and a set of whitelist functions **Whitelists** each a predictive model specified as a probabilistic program, our goal is to then create noising/obfuscation functions, that $\forall B \in$ **Blacklists** satisfies the security policy $P(B(\delta^{\sim}_{2}))$, while $\exists W \in$ **Whitelists** satisfies the utility policy $U(W(\delta^{\sim}_{2}))$. In [1] we demonstrated one means of learning obfuscation functions that satisfy security and utility policies in sharing with Bob an obfuscated version of Alice's secret value, $Q = O(s)$, via suppressing and selecting to share only particular features of Alice's secret information.

References

[1] Supriyo Chakraborty, Kasturi R. Raghavan, Mani B. Srivastava, Chatschik Bisdikian, and Lance M. Kaplan. Balancing value and risk in information sharing through obfuscation. In *Proceedings of the International Conerence on Information Fusion (FUSION)*, 2012.

[2] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying information flow with beliefs. *Journal of Computer Security*, 17(5):655–701, 2009.

[3] Andrew D. Gordon, Mihhail Aizatulin, Johannes Borgstroem, Guillaume Claret, Thore Graepel, Aditya Nori, Sriram Rajamani, and Claudio Russo. A model-learner pattern for bayesian reasoning. In *Proceedings of the ACM SIGPLAN Conference on Principles of Programming Languages (POPL)*, 2013.

[4] Piotr Mardziel, Michael Hicks, Jonathan Katz, and Mudhakar Srivatsa. Knowledge-oriented secure multiparty computation. In *Proceedings of the ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS)*, 2012.

[5] Piotr Mardziel, Stephen Magill, Michael Hicks, and Mudhakar Srivatsa. Dynamic enforcement of knowledge-based security policies. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, 2011.

[6] Geoffrey Smith. On the foundations of quantitative information flow. In *Proceedings of the Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, 2009.