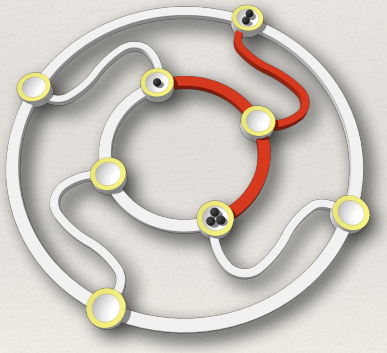
Quantitative information flow:

when are partial breaches of confidentiality insignificant?

Annabelle McIver



Workshop on software correctness
and reliability

A brief history

- 1970's Statistical databases — Census "daticians" worried about privacy breaches in published census data.
- 1982 Goguen and Meseguer — Non-interference security: sought to tease out what it means for confidential information to leak.
- 1993 Landauer — Lattice of Information. A first attempt to compare the leakiness of security mechanisms.
- 1995-ish side channel attacks which undermined security protocols especially when implemented on small devices.
- 2000's Quantitative information flow — measure information loss using Shannon Entropy; "Channel model of information flow" for quantitative non-interference.
- 2009 Min-Entropy, Guessing Entropy — these are better for measuring information loss damaging to security.
- 2012 "Generalised Entropy" to define threat models; security refinement.
- 2014 Generalised capacity to prove that any leak is "small enough".

Secrets

We model a secret as something that an attacker can try to guess.

$\mathcal{X}$	Basic type for secrets
$\mathbb{D}\mathcal{X}$	Probability distributions over secrets

The distribution over a secret models how well hidden the secret is.

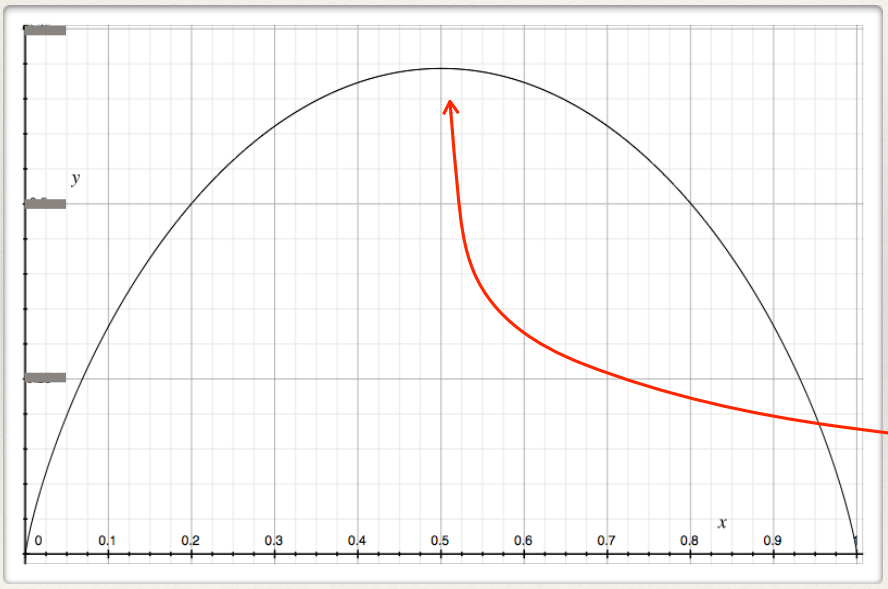
A uniform distribution means that the attacker knows nothing about the secret except for its type.

A uniform distribution over some subset Y means that the attacker can rule out the complement $\mathcal{X} \setminus Y$.

Entropy

We can get a better handle on how well hidden is the secret by using "Entropy"

H(X)



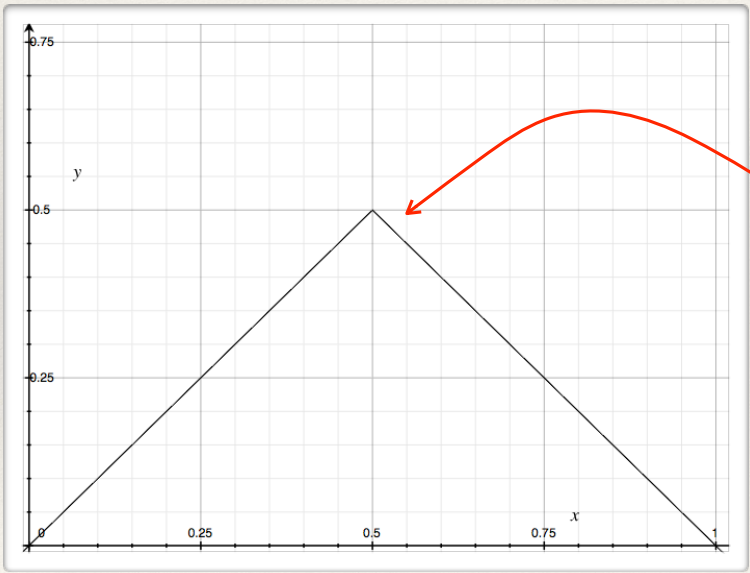
Probability of X

For example, Shannon Entropy is one way to measure "uncertainty" (wrt the attacker's point of view).

The higher the entropy
the more uncertainty
because the more resources are required.

Bayes Risk

We can get a better handle on how hidden the secret is by using "Entropy"



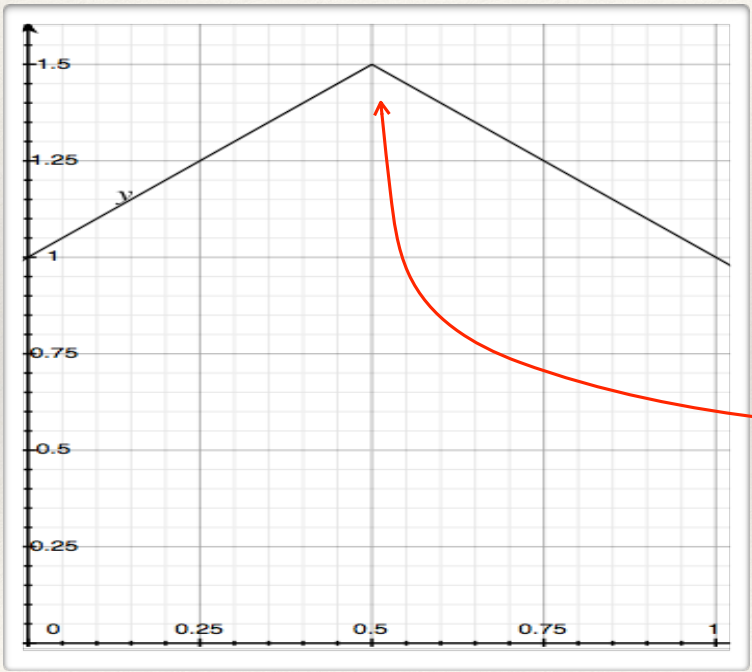
Probability of X

Bayes Risk gives the average over "$1 fines" to the attacker if he incorrectly guesses the secret in one try.

The higher the entropy the more uncertainty because the more the attacker is fined on average.

Guessing Entropy

We can get a better handle on how disguised is the secret by using "Entropy"

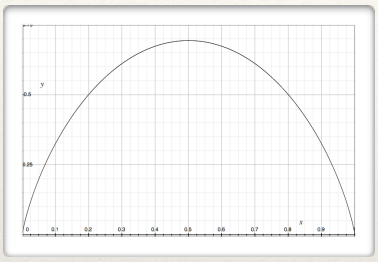


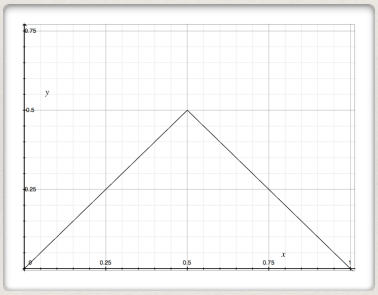
Probability of X

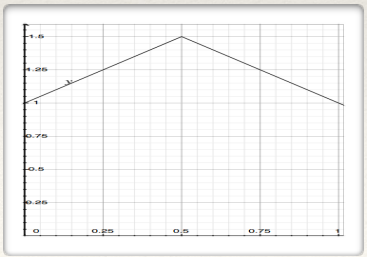
Guessing Entropy gives the expected number of guesses it takes for the attacker to guess the entire secret.

The higher the entropy the more uncertainty because the more guesses on average are needed to uncover the secret.

Abstract Properties of Entropy

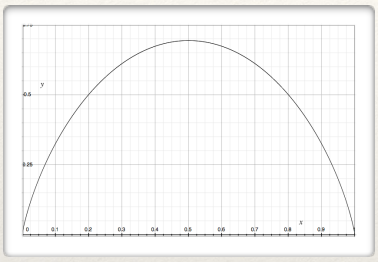


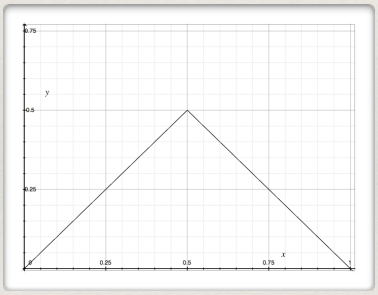


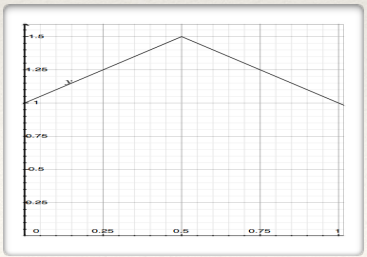


- Different secrets can have different associated risks, depending on the context.
- Indistinguishability between secrets: secrets which are hard to distinguish in probability should be hard to distinguish in entropy.
- Averaging between uncertainties increases the overall uncertainty.

Abstract Properties of Entropy







- Entropy is a function from secrets to reals.

$$\mathbb{D}\mathcal{X} \to \mathbb{R}$$

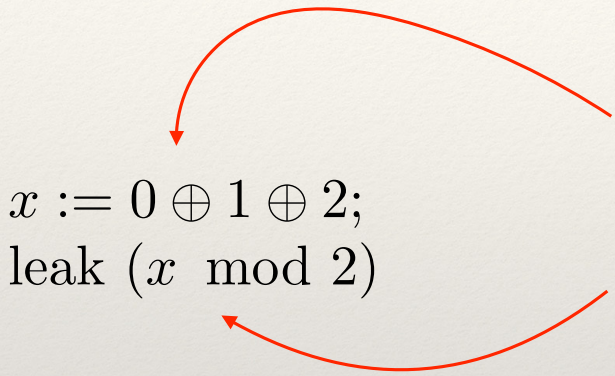
- Entropy is continuous.

$$\delta_1 \to \delta_2 \quad \Rightarrow \quad U(\delta_1) \to U(\delta_2)$$

- Entropy is concave.

$$U(\delta_1 \oplus_p \delta_2) \;\geq\; p \times U(\delta_1) + (1-p) \times U(\delta_2)$$

Mechanisms change the risk

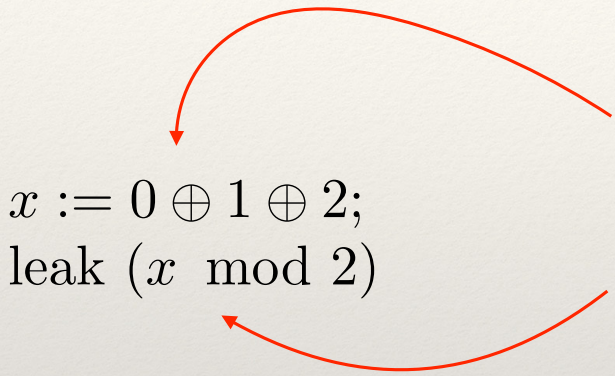


Initially the secret x is uniformly set over 0, 1, 2.

Subsequently its low order bit is revealed.

What does this leak tell the attacker about the secret?

Mechanisms change the risk



Initially the secret x is uniformly set over 0, 1, 2.

Subsequently its low order bit is revealed.

What does this leak tell the attacker about the secret?

If “1” is leaked then the attacker deduces that the secret x *is* 1;
If “0” is leaked then the attacker deduces that the secret *is not* 1, but is uniformly distributed over 0 or 2.

Generalising Mutual Information

The theory of Shannon Information introduces the idea of "mutual information" between random variables X and Y. Mutual information measures the "mutual dependence" between X and Y.

$$x := 0 \oplus 1 \oplus 2;$$
$$\text{leak } (x \mod 2)$$

Shannon based: $\log 3 - 2/3 \log 2 \approx 0.918$

Bayes Risk based: $2/3 - 1/3 = 1/3$

Guessing Entropy based: $2 - 4/3 = 2/3$

These measure actual "dollar" amounts in reduction in "costs" to the attacker.

Measure the change in Entropy

Shannon: Before

$$H[x := 0 \oplus 1 \oplus 2] = \log 3$$

Shannon: After

$$H[x := 0 \oplus 1 \oplus 2; \text{“leak } (x \mod 2)\text{”}] = \frac{2}{3} \log 2$$

Bayes Risk: Before

$$B[x := 0 \oplus 1 \oplus 2] = \frac{2}{3}$$

Bayes Risk: After

$$B[x := 0 \oplus 1 \oplus 2; \text{“leak } (x \mod 2)\text{”}] = \frac{1}{3}$$

Guessing Entropy: Before

$$G[x := 0 \oplus 1 \oplus 2] = 2$$

Guessing Entropy: After

$$G[x := 0 \oplus 1 \oplus 2; \text{“leak } (x \mod 2)\text{”}] = \frac{4}{3}$$

Generalised entropy and attack strategies

Loss functions and attack strategies

A loss function ℓ is a mapping , $\mathcal{W} \to \mathcal{X} \to \mathbb{R}$

Where $\mathcal{W}$ is a (finite) set of strategies.

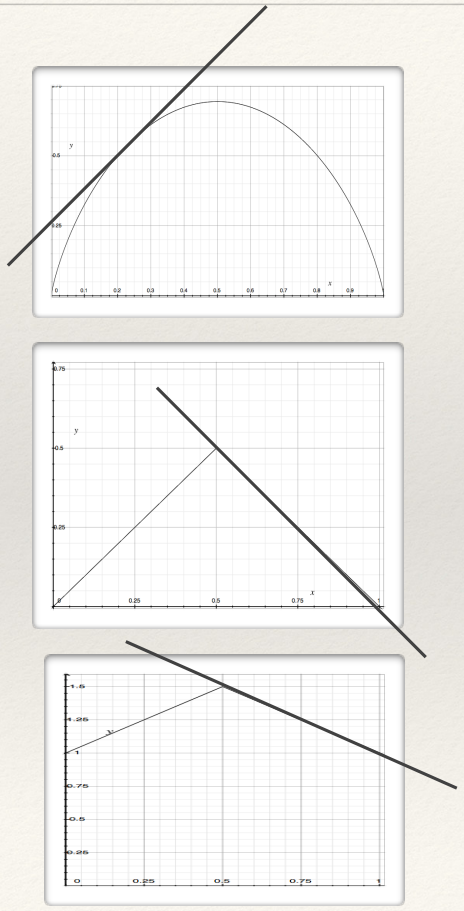
Given a loss function ℓ and a distribution representing some secret π

We define the expected loss as

$$U_\ell(\pi) \quad := \quad \min_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \ell.w.x \times \pi_x$$

Each of the entropies above can be expressed as a U_ℓ for some ℓ

Abstract Properties of Entropy



- For fixed w The function $\ell.w$ is the cost function for strategy
- Considered as a function from secrets to reals $\ell.w$ becomes a tangent to the entropy curve.

$$\mathbb{D}\mathcal{X} \to \mathbb{R}$$

- Entropy is then determined exactly by the set of attacker strategies.

Loss functions and attack strategies

We interpret the "generalised" entropies as costs related to attacker strategies.

Bayes Risk - The attacker has three strategies:

A: Guess "x=0" costs $1 if the guess is wrong;
B: Guess "x=1" costs $1 if the guess is wrong;
C: Guess "x=2" costs $1 if the guess is wrong;

Before the leak, for a uniform prior, any of the three choices costs him the same, i.e. 2/3.

$$x := 0 \oplus 1 \oplus 2;$$
$$\text{leak } (x \mod 2)$$

Change of strategy: If he observes "1" he uses B. If he observes "0" then he uses either A or C. The average cost related to this strategy is zero 1/3 of the time, and only 1/2 for 2/3 of the time giving a total of 1/3.

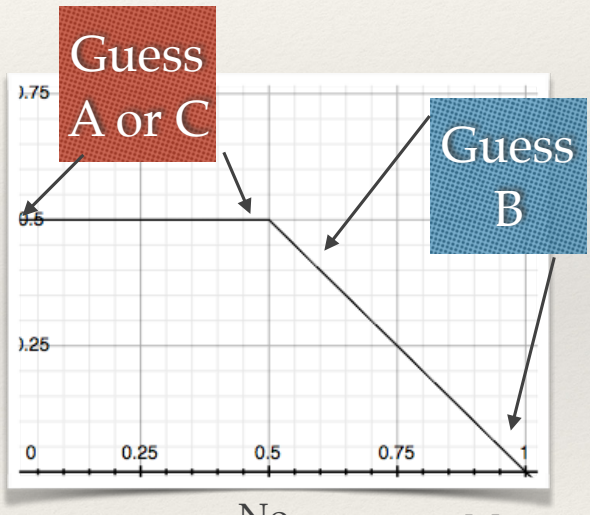
Before

We interpret the “generalised” entropies as costs related to attacker strategies.

Bayes Risk - The attacker has three possible choices to build guessing attacks:

A: Guess “x=0” costs $1 if the guess is wrong;
B: Guess “x=1” costs $1 if the guess is wrong;
C: Guess “x=2” costs $1 if the guess is wrong;

$$x := 0 \oplus 1 \oplus 2;$$
$$\text{leak } (x \mod 2)$$



Less likely that x =1

No clue

More likely that x =1

Consider the strategy:
Randomly select A or C, unless the prior indicates it’s more likely the secret is 1. In that case select B.

After

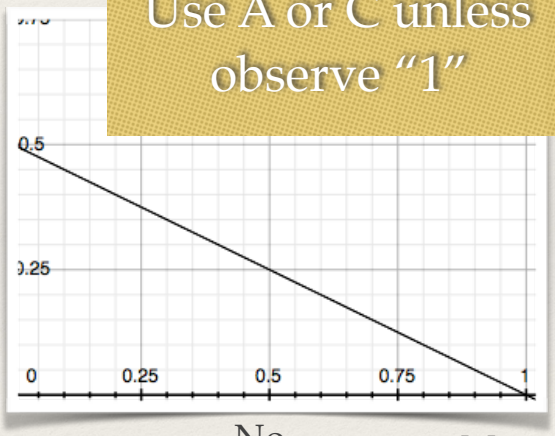
We interpret the "generalised" entropies as costs related to attacker strategies.

Bayes Risk - The attacker has three possible choices to build guessing attacks:

A: Guess "x=0" costs $1 if the guess is wrong;
B: Guess "x=1" costs $1 if the guess is wrong;
C: Guess "x=2" costs $1 if the guess is wrong;

$$x := 0 \oplus 1 \oplus 2;$$
$$\text{leak } (x \mod 2)$$

Change the strategy based on the observations.

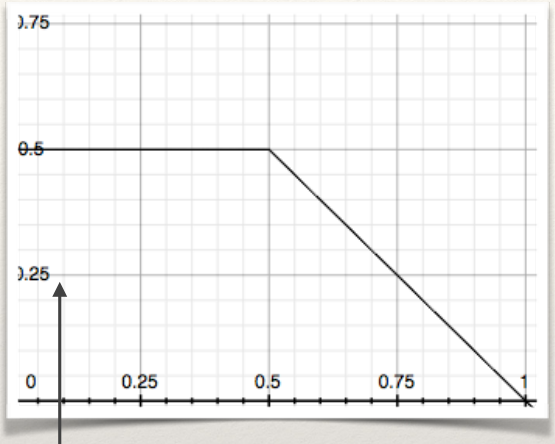


Less likely that x =1

No clue

More likely that x =1

Attacking after is more worthwhile than attacking before

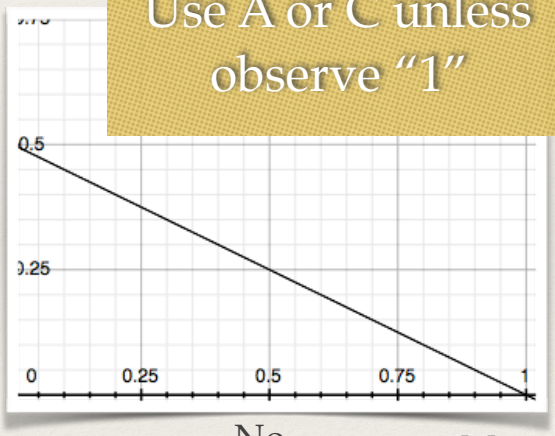


$x := 0 \oplus 1 \oplus 2;$

$\text{leak } (x \mod 2)$

He can always find a strategy that, will reduce his losses if he attacks after the leak.

Use A or C unless observe "1"

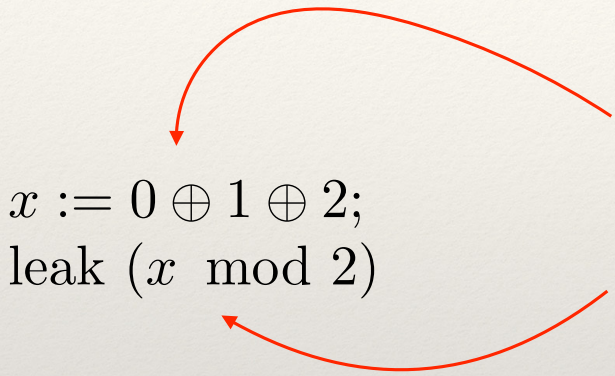


Less likely that x =1

No clue

More likely that x =1

Change of entropy is a measure of effectiveness of strategy change



Initially the secret x is uniformly set over 0, 1, 2.

After the leak we take the observations into account.

"1" is leaked with probability 1/3 — in this case the attacker knows the secret to be 1.

"0" is leaked with probability 2/3 — in this case he deduces that the remaining uncertainty is a *posterior distribution* evenly split between 0 and 2.

Risk after the leak

The posterior distributions are calculated using Bayes' Formula:

Pr(secret is x, Given observation A)
= Pr(secret is x AND observation is A)/Pr(observation is A).

The probabilities are worked out from the joint probability distribution generated from the prior and the observations.

The posterior average loss with respect to loss function is

$$\sum_{a \in \mathcal{A}} pr(a) \times U_\ell(\pi^a)$$

Concavity explains why the posterior average entropy is reduced.

where $pr(a)$ is the probability of observing a and π^a is the posterior distribution given a

Can one mechanism be more secure than another?

Robust comparisons

Given two medical devices for testing for some disease, both taking inputs X from patients, we can decide how good these devices are by determining which is better "on average" in identifying patients who have the disease or not. The secret is whether the patient actually has the disease or not; the leak is whether the observation decreases the original uncertainty.

Our theory tells us that if we can model the information leaks as *stochastic channels*, from secrets to observations then we can make some robust judgements about their comparative security properties.

Robust comparisons: refinement for security

The question is then whether there is a *robust* leakage ordering, allowing us to conclude that one channel *never* leaks more than another, regardless of the prior or gain function. Such a robust ordering could support a stepwise refinement methodology for constructing secure systems.

> *Definition 5.1:* B *composition refines* A, written $A \sqsubseteq_\circ B$, if there exists a channel C such that $B = AC$.

The main interest in composition refinement is its relation to g-leakage. First, composition refinement implies a strong g-leakage ordering; this can be seen as an analogue of the classic *data-processing inequality*.

> *Theorem 5.1 (Data-processing inequality):* If $A \sqsubseteq_\circ B$ then the g-leakage of B never exceeds that of A, for any prior π and any loss function g.

Applying the theory

There are many ways to apply these results on information flow. Here are three of them, some of which we have tried.

1. Evaluating the overall information leaks in programs; *Side channel analysis*.
2. Information flow in security protocols; *Voting protocols*.
3. Developing reasoning techniques based on *refinement* to explain and verify how to use security protocols and to determine that elaborate algorithms meet their information-flow requirements.

Voting protocols

A basic principle of democratic voting is *privacy*.

A second principles is *integrity* which means that votes must be accurately counted, i.e. so that the result is a "true" amalgamation of the votes cast.

There is an obvious tension between privacy and integrity.

We wanted to study that tension from an *information flow* perspective, in particular:

Given that some information must flow (eg the winners/losers have to be announced), can we determine how much privacy that gives voters?

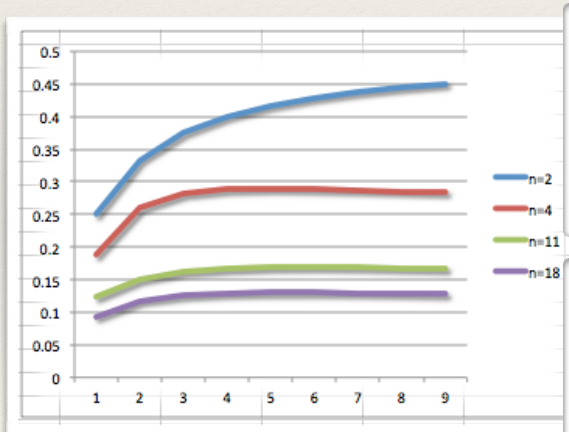
Voting protocols: summary results

1. The privacy questions we investigated using loss functions are:
   (a) What is the chance that an observer can correctly guess a voter's cast ballot?
   (b) What is the expected number of voters' cast ballots can be guessed correctly?
2. We first looked at "first past the post" voting protocols, and compared two versions: the first where *only* the winner is announced, and the second where the *actual counts* are also announced. We found that there is *no difference* in privacy wrt the two privacy scenarios (a) and (b).
3. The chance of guessing any voter's ballot *as a direct result of the information flow* decreases (tends to 0) as the pool of voters increases (gets very large).
4. We next looked at Instant Run-Off elections. These are slightly simpler than Australia's full preferential counting, but the information flow is similar. In fact they are a special case of first past the post with multiple candidates.

Voting protocols: summary results

We next looked at Instant Run-Off elections. These are slightly simpler than Australia's full preferential counting, but the information flow is similar. In fact they are a special case of first past the post with multiple candidates.

Increased risk that vote can be identified.



Number of possible observations.

For 2 voters preferential voting increases the risk by 45%

For 18 voters, the risk is only increased by 12%, and this decreases the larger the voting population.

Channel capacities

Channel capacities give robust upper bounds over all priors and loss functions.

Maximum taken over loss functions with range in [0, 1], and all possible priors.

Multiplicative capacity $$\max_{\ell,\pi} \frac{(1 - U_\ell[\pi, C])}{(1 - U_\ell[\pi])}$$

Additive capacity $$\max_{\ell,\pi} (U_\ell[\pi] - U_\ell[\pi, C])$$

Channel capacities

Multiplicative capacity (m):

- Easy to calculate
- log(m) provides an **upper bound** on average bits leaked (as in Shannon leakage).

$$a \leq 1 - 1/m$$

Additive capacity (a):

- Easy to calculate for fixed prior

(but not for all priors)

- Gives an upper bound on actual cost to the attacker

Equality is achieved for channels that leak information "deterministically" rather than unpredictably.

Side channel analysis of code

```
// B for base, the cleartext; E for exponent, the key: precondition is B,E >= 0,0 .
// P for power, the ciphertext.
P:= 1
while E!=0  // Invariant is P*(B^E) = b^e, where b,e are initial values of B,E .
   D:∈ [2,3,5]              // D for divisor; uniform choice from {2,3,5}.
   R:= E mod D;             // R for remainder.
   if R!=0 then P:= P*B^R fi // Side-channel : is E divisible exactly by D ?
   B:= B^D                  // D is small: assume no side-channel here.
   E:= E div D              // State update of E here: again no side-channel.
end
// Now P=b^e and E=0: but what has an adversary learned about the initial e ?
```

This program calculates P:= b^e where b, e are the initial values of base and (private key) exponent, initialised outside the program, and P is the result, set by the program itself.

A side-channel attacker detects whether the **then**- or the **else**-branch of the indicated **if** is taken — it corresponds to **leak** R!=0. If D were always 2, then that would reveal the bits of the private key E, initially e, one-by-one. To mitigate that, the divisor D is instead chosen uniformly from some small set, here $\{2, 3, 5\}$, afresh on each iteration.

Fig. 4. Defence against side-channel analysis in exponentiation [30]

Side channel analysis of code

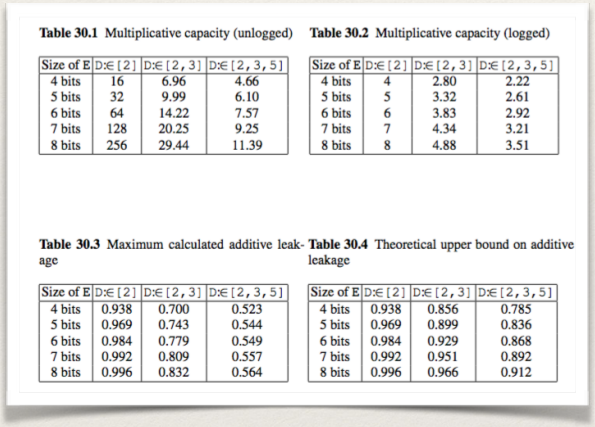


Table 30.1 Multiplicative capacity (unlogged)

Size of E	D:∈[2]	D:∈[2,3]	D:∈[2,3,5]
4 bits	16	6.96	4.66
5 bits	32	9.99	6.10
6 bits	64	14.22	7.57
7 bits	128	20.25	9.25
8 bits	256	29.44	11.39

Table 30.2 Multiplicative capacity (logged)

Size of E	D:∈[2]	D:∈[2,3]	D:∈[2,3,5]
4 bits	4	2.80	2.22
5 bits	5	3.32	2.61
6 bits	6	3.83	2.92
7 bits	7	4.34	3.21
8 bits	8	4.88	3.51

Table 30.3 Maximum calculated additive leakage

Size of E	D:∈[2]	D:∈[2,3]	D:∈[2,3,5]
4 bits	0.938	0.700	0.523
5 bits	0.969	0.743	0.544
6 bits	0.984	0.779	0.549
7 bits	0.992	0.809	0.557
8 bits	0.996	0.832	0.564

Table 30.4 Theoretical upper bound on additive leakage

Size of E	D:∈[2]	D:∈[2,3]	D:∈[2,3,5]
4 bits	0.938	0.856	0.785
5 bits	0.969	0.899	0.836
6 bits	0.984	0.929	0.868
7 bits	0.992	0.951	0.892
8 bits	0.996	0.966	0.912

These figures were calculated in Haskell by using Haskell's Probability Monad standard library, and giving a monadic semantics to "leak" statements as well as state updates. Program semantics is greatly simplified using Monads rather than matrices, giving programs type $\mathbb{D}\mathcal{X} \to \mathbb{D}^2\mathcal{X}$

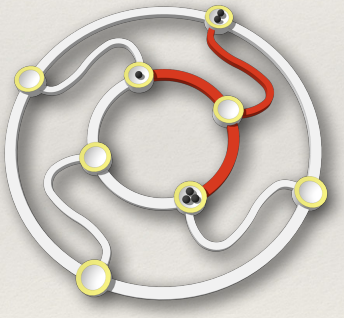
Calculations produced by Carroll Morgan.

Conclusions

- Quantitative information flow gives the ability to define relevant scenarios using loss functions;
- We can define a compositional refinement order using this idea: refinement of P by Q means that by any measure of entropy, Q leaks less than does P. This relation is robust even when P and Q are placed in context.
- Applying these concepts in real situations means that we can make robust comparisons e.g. information flow in voting.
- We can develop source-level reasoning to show that protocols leak no more information than do minimal specifications which describe only “logical leaks”.
- We can develop tools which measure actual information leaks which can be used when the channel models are too large to inspect directly.

Quantitative information flow

Annabelle McIver



Reliability
Workshop 2016