TPDP, WPES and CCS talks

Sebastian Meiser[1]
1: University College London, United Kingdom, e-mail: s.meiser@ucl.ac.uk

October 17, 2018

Abstract

This is a document of summaries of talks I attended at CCS 2018. Please note that in most cases I have not read the paper before writing the summary, so my summaries might be very shallow and sometimes factually incorrect. If you notice any inconsistencies, errors or would like me to include aspects that I've missed, please just send me an email.

Contents

1 Pre-Conference Workshops

1.1 Invited talk TPDP: Composition, Verification, and Differential Privacy

Speaker: Justin Hsu

After introducing differential privacy in general, we see a few well-known composition results. Post-processing is seen as an instantiation of sequential composition of an (ε, δ)-DP mechanism with a $(0, 0)$-DP mechanism, which is kind of nice. Similarly, local DP is presented as an instantiation of parallel composition.

1.1.1 Formally verifying privacy

Our goals here are twofold: we want to have *dynamic* verification, i.e.,raise errors if DP is violated and *static* verification, i.e., checking DP on all possible inputs. As a side-note, we want to simplify verification by using composition results; composition allows verification techniques to have much better automation.

The Fuzz family of languages allows for describing each type with a metric and, using group privacy ($(\varepsilon, 0)$-DP for $\Delta_f = 1$ implies $(k \cdot \varepsilon, 0)$-DP for $\Delta_f = k$). The description allows for an immediate static analysis that can use both sequential and parallel composition. These languages, so far, cannot deal well with ADP, but the speaker is currently working on that.

Privacy as an approximate coupling Idea: verify privacy by game hops. We somehow relate pairs of sampling instructions to show properties of a pair of inputs, but I'm not sure how exactly we do that. This static-analysis technique seems to be related to how Tim and me modeled differential indistinguishability, i.e., as a property of a pair of programs/machines. They, so far, cannot automatically generate proofs, but can check them.

1.1.2 Advanced composition

They analyze the old "advanced composition theorem" by Dwork, Rothblum and Vadhan. The speaker says that the analysis is more complicated, since the composition theorem needs to be applied "as a block", instead of on small units of the program. The speaker then mentions that novel approaches (e.g., by Mironov on RDP) makes formal verification easier, as it allows to analyze a program step-by-step. I think that using our privacy buckets approach, such an analysis could be improved even more and made flexible as well.

1.2 Modularity

The verification approach aims at being able to modularly compose differential privacy mechanisms. Their approach does not solve a fundamental problem that black-box composition of DP mechanisms have: after black-box composition there is potentially too many points where noise is added. So, a more thorough approach would divide DP mechanisms into summarizing data (e.g., via statistical queries) and adding noise. Then, composition would compose the summarization operations and then only add noise once. Such an approach would deteriorate utility much less.

1.3 Invited Talk TPDP 2: Deploying Differential Privacy for Learning on Sensitive Data

Speaker: Ulfar Erlingsson

The speaker mentions problems with RAPPOR, mostly that for the quantities of data they require, the privacy guarantees are deteriorating too fast. The main problem seems to be that they have to use local DP mechanisms.

The aim of their Prochlo realization is to combine the local differential privacy mechanisms with some central DP approach.

They notice that if they don't need correlations between data points from the same user, anonymous communication can help them. As a result, they have a partially central approach. They group data based on useful features.

Their approach combines three aspects of uncertainty: the uncertainty introduced by the local DP mechanism (via randomized response), the uncertainty introduced by the combination (anonymity and random shuffling) and the uncertainty applied to the results afterwards (in a centralized DP manner).

1.3.1 Private neural networks

The speaker identifies the problem of machine learning models, e.g., NNs, memoize the training data. This confirms known results on adversarial ML and works by Dawn Song and Vitaly Shmatikov.

Specifically, they introduce canaries, i.e., specific numbers, and want to check whether the NN is surprised to see this canary or not. In other words, they check whether the NN learnt this canary. Their experiments show that NNs indeed learn such canaries. Learning the canaries seems to reach its worst case early, with the convergence of the model.

The speaker emphasizes that this memoization is different from overfitting, because in spite of memoization generalization works well.

The speaker emphasizes that there are edge cases that the NNs seemingly have to memorize exceptions, because there does not seem to be a rule to characterize these edge cases. These exceptions are not necessarily important for the final model, as they contradict the general paradigm of machine learning: to learn general properties of samples and not exceptions. This makes differential privacy a natural fit for ML.

We now get an introduction into the privacy-preserving stochastic gradient descent (SGD) [**?**] work and the PATE paper [**?**].

In their case study, they use prior knowledge and use a strong bayesian prior.

1.4 Local differential privacy for evolving data

Speaker: Matthew Joseph, collaboration with Aaron Roth, Jonathan Ullman and Bo Waggoner
Paper: `https://arxiv.org/pdf/1802.07128.pdf`

The main difficulty they tackle is evolving data, i.e., data that can change over time. The simplest solution in terms of differential privacy would be to simply have a randomized response for every user in every round. While this is very simple, privacy deteriorates over time. To moderately improve it, you can apply subsampling, i.e., ask different subsets of users every time. This is better in terms of privacy, but the error is still large. Local sparse vectors (?) would allow for nice privacy guarantees, but require an exponentially larger number of samples.

The solution lets users "vote" to change statistics, i.e., they construct an adaptive mechanism.

In each epoch t, each user creates a localized bit, then generates a vote for or against updating and then depending on the average vote the analyst decides whether or not the estimate should be updated (requiring more interaction) or not.

Each user can guess whether or not the global estimate has changed significantly. We aren't completely sure on the maths here, but apparently each user locally simulates what the estimate should be (in the current round) and compare that to the previous global estimate. If these don't differ much, the user is content; if they do differ significantly, the user votes for a change.

If a change occurred, the analyst collects the changed estimates from the users to update the global estimate. Otherwise the analyst just keeps the previous estimate.

I'm not completely convinced yet by the proof intuition for why their mechanism satisfies differential privacy; a closer look into their paper [1] might be in order to understand what exactly they are doing and why that is privacy-preserving, particularly towards the data analyst that collects the votes and combines the estimates.

1.5 WPES: TightRope: Towards Optimal Load-balancing of Paths in Anonymous Networks

Speaker: Hussein Darir (University of Illinois at Urbana-Champaign), collaboration with Hussein Sibai (University of Illinois at Urbana-Champaign), Nikita Borisov (University of Illinois at Urbana-Champaign), Geir Dullerud (University of Illinois at Urbana-Champaign), and Sayan Mitra (University of

[1] available online: `https://arxiv.org/pdf/1802.07128.pdf`

Illinois at Urbana-Champaign)

The paper is about Tor; some Tor circuits tend to be slow. Knowing the current state of the network (in terms of congestion and usage) leaks too much information, but maybe differential privacy can help us here. The goal is to use load balancing with locally optimal mechanisms and this mechanism should then be differentially private.

They assume that each user only holds a single static circuiy that is active all the time.

Each relay computes the ratio between their capacity over the number of paths going through it. Then they choose the relay with the minimal ratio, called the bottleneck relay (it is most congested). The path(s) going through the bottleneck relay are tagged with the ratio of the bottleneck relay. Then from the other relays, the capacity used by the path through the bottleneck relay is subtracted from these relays' capacities, and then this path is removed. The algorithm then is repeated to compute the bandwidth available to each path. They evaluate this and find (unsurprisingly) that the bandwidths are unfairly distributed over paths.

Their (non-private) algorithm now takes the existing bandwidth capacities and ratios into account and also computes the ratio where one more path is added to the relay. They find the most congested node, i.e., the one where the ratio would be worst if we added one. We then remove this relay (and paths through it; and update the ratios). This process is repeated until we can build a path. Using this technique makes path selection much more fair, but violates privacy.

To achieve differential privacy, they need up-to-date statistics. I'm not completely sure how they achieve that; they seem to sample many times, i.e., have each user locally simulate how the network might have changed? I'm not quite sure what exactly they do and what privacy guarantees they achieve, but their evaluation shows that still some improvement over a simple random choice (without optimization) is possible.

They did not analyze the impact their work has on the anonymity provided by Tor; juts the impact of releasing their histogram.

1.6 WPES: ClaimChain: Improving the Security and Privacy of In-band Key Distribution for Messaging

Speaker: Wouter Lueks (Ecole Polytechnique Fédérale de Lausanne), collaboration with Bogdan Kulynych (Ecole Polytechnique Fédérale de Lausanne), Marios Isaakidis (University College London), George Danezis (University College London), and Carmela Troncoso (Ecole Polytechnique Fédérale de Lausanne)
Paper: `https://arxiv.org/abs/1707.06279`

The classical PKI problem is: ow does Alice find Bob's key? They analyze how to distribute keys for an email structure that allows for encrypted communication. Whenever someone sends an email, they want to attach some small datastructure that includes claims about their own key and about their friends' keys. Moreover, they want to hide whose keys they are communicating from unauthorized people. They intend tp do that by adding some access control mechanism, specifying who can access these records. Finally, they want to get non-equivocation, i.e., sending around different keys supposedly belonging to the same entity, accessible by different people.

Their Approach They store all claims (of the form "entity, key") into an encrypted dictionary. To hide the indexes of the dictionary, they put a verifiable random function in there, i.e., some apparently random value; they communicate this value to people of interest. They then put a proof that they computed the VRF correctly into the encrypted part. They then chain these claims together with a hash chain. This basically is the ClaimChain that can be attached to emails.

1.7 WPES: What's a little leakage between friends? (Short)

Speaker: Sebastian Angel, collaboration with David Lazar, Ioanna Tzialla
Paper: `https://arxiv.org/abs/1809.00111`

Metadata-private messaging systems allow communication without leaking metadata to providers, servers or users not involved in the communication. In the simplest case, everyone sends packets to everyone else. Existing MPM systems avoid this broadcast by limiting the number of messages per round. Clients now

have to start scheduling their own communication (and potentially wait until their messages are being sent). To actually communicate, you can have a dialing round in which you negotiate a communication round.

They then analyze what happens if a malicious user tries to gain information about whether others are talking: the idea is that an adversary can ask a user whether they want to talk; since every user only has a limited number of (real) messages per round, they can see whether the user is "free" or "already in conversation". To counter this, they envision a private answering machine, that hides from callers whether or not a user is talking; moreover eventually a caller needs to get through and it should be efficient, i.e., the capacity of the answering machine should be significantly smaller than full broadcast. Their proposal has several limitations, including that everyone needs to have a max number of friends m; then statically map callers to rounds (mod m). Drawbacks are potential limitation and leakage of the number of friends and an increased latency (if the answering machine has low capacity or the user many friends).

1.8 WPES: DynaFlow: An Efficient Website Fingerprinting Defense Based on Dynamically-Adjusting Flows (Short)

Speaker: David Lu (MIT PRIMES), collaboration with Sanjit Bhat (MIT PRIMES), Albert Kwon (MIT), and Srinivas Devadas (MIT)
Paper: `https://dl.acm.org/citation.cfm?id=3268960`

The paper aims at countering website fingerprinting, particularly when people are using Tor. As we know, Tor is vulnerable to traffic analysis attacks, including website fingerprinting attacks. The talk considers an open-world scenario for fingerprinting. Existing defenses, e.g., supersequence defenses over-approximate all websites within an anonymity set and make all these websites look the same. Similarly, constant flow defenses enforce a constant transmission rate, which introduces a significant overhead as well.

Their approach morphs traffic into fixed bursts: o outgoing and i incoming packets, for fixed values o and i. Moreover, they vary the inter-packet timing interval: t changes every b bursts, t is chosen from some set $\{t_1, \ldots, t_k\}$ and up to a adjustments are allowed.[2]

They evaluate their approach against a couple of existing attacks, using both a "medium security" and a "high security" setting. Their results look promising: the overheads are only 28% of time overhead and about 110% of bandwidth overhead.

2 CCS, Tuesday, Privacy Session

2.1 ABY3: A Mixed Protocol Framework for Machine Learning

Authors: Payman Mohassel (Visa), Peter Rindal (Oregon State University)
Paper: `https://eprint.iacr.org/2018/403.pdf`

Three party SMPC; each party encrypts their data and sends some of their shares to the other parties. They use three types of sharing: arithmetic (sums; $x = x_1 + x_2 + x_3$), binary (xor $x = x_1 \oplus x_2 \oplus x_3$) and Yao's garbled circuits.

Their main focus is on machine learning, for which they represent floating point numbers as a pair of integers that they process separately, which is pretty straight-forward. They use a trick in which they briefly fall back to a 2-out-of-2 secret sharing mechanism for performing multiplications securely without adding a lot of overhead. Moreover, they can deal with matrix multiplication with less communication overhead than state-of-the-art.

They show a general way to convert freely between arithmetic secret sharing, boolean secret sharing, and Yao's garbled circuits.

Finally, they can perform linear and logistic regression as an SMPC and repeat the process to train neural networks. Particularly for neural networks, their performance is orders of magnitude better than previous work on two-party SMPC (called "SecureML").

[2] I'm not sure within which time frame a is measured

2.2 Voting: you can't have privacy without verifiability

Speaker: Joseph Lallemand (CNRS, Inria, Loria, Université de Lorraine, France), collaboration with Véronique Cortier (CNRS, Inria, Loria, Université de Lorraine, France)
Paper: `https://hal.inria.fr/hal-01858034/document`

As the title suggests, the paper is about the formal verification of voting. The speaker emphasizes that, particularly in eVoting, there are many potential adversarial parties and attack angles, including dishonest voters, ballot boxes, tallying authorities and communication channels. The main gals they have are privacy (of votes, as an indistinguishability property), verifiability (voters can check that their votes have been correctly cast, including: individuals checking that the vote is in the box, everyone checking that the results have been computed based on the votes in the box and finally, the verification that only valid voters participated).

Privacy and verifiability naturally is contradictory: it is quite easy to achieve one without the other, but hard to achieve both together.

They show, in fact, that their definition of privacy directly implies their definition of individual verifiability. I presume that they implicitly require correctness, as otherwise this is clearly not true.[3] They define individual verifiability as the inability of the attacker to modify the result of the voting: The adversary is allowed to cast votes, but must not be able to "remove the honest votes from the result". I'm not sure what exactly that means, but I think that this is where their implicit correctness assumption comes into play.

They then prove that an attacker that can manipulate individual verifiability (i.e., remove people's votes) can break privacy. The proof starts with an attacker that can manipulate people's votes and they then use this attacker to break privacy (just fix Alice's vote as either 0 or 1) and then, since the **distributions** the adversary has to provide for privacy have to **return the same result** for the voting process, the adversary can learn Alice's vote by checking whether the tally changed. This proof technique is tailored to the attacker, but in their paper they generalize the technique to work with any attacker.

Learning from their insight, they then define a novel privacy definition based on the verification steps of the protocol. The ballot-box is now dishonest. The attacker is given access to an oracle that lets people verify their votes. The attacker can only see the results after all voters have verified their votes. Their implication still holds.

3 CCS, Tuesday, Differential Privacy 2 session

3.1 Preserving Both Privacy and Utility in Network Trace Anonymization

Speaker: Meisam Mohammady, collaboration with Lingyu Wang (Concordia institute for information systems engineering), Yuan Hong (Illinois Institute of Technology), Habib Louafi (Ericsson Research Security), Makan Pourzandi (Ericsson Research Security), Mourad Debbabi (Concordia institute for information systems engineering)

Outsourcing network traces is relevant for getting top security monitoring and analytics. Sending network traces to some outsider obviously comes with privacy concerns. The author mentions an anonymization function: the existing prefix preserving anonymization function preserves prefix equality, but has a variety of vulnerabilities. One of these vulnerabilities includes simply injecting a few traces to then find traces for similar subnets.

The adversary is honest-but-curious and tries to find all possible matches between the anonymized and the original traces. Moreover the adversary has α-knowledge, i.e., can have successfully injected α traces.

They send several traces to the analyst and somehow hide the real one in there..? They first hide the original trace, then partition the ptrace, then encrypt each partition a different number of times (thus each partition is prefix-preserved, but globally it isn't). They create several views then, one of which is the real one (why does that not have the same problem again?). To protect against such madness, they require their fake views to be within a e^{ε} privacy loss of the real view, i.e., they all could have been "real". I'm not quite

[3] Consider the protocol that always outputs the same "results" independent of the votes; this preserves privacy, but since the votes aren't used, they cannot be verified.

sure how they achieve that, but it involves a lot of encrypting and "reverse encrypting", which might be decryption.

3.2 Toward Detecting Violations of Differential Privacy

Speaker: Yuxin Wang (Pennsylvania State Univerisity), collaboration with Ding Ding (Pennsylvania State Univerisity), Guanhong Wang (Pennsylvania State Univerisity), Danfeng Zhang (Pennsylvania State Univerisity), Daniel Kifer (Pennsylvania State Univerisity)

The aim of the talk is to test the design of algorithms to find out whether they are differentially private. To this end, they present a tool. They use pure DP in their work. Their aim is to find a good counterexample, i.e., a pair of adjacent databases and a set S s.t. the DP formula is violated. That sounds quite straightforward, but I'm not sure how easy it is to find these counterexamples.

- First they need to find candidate databases; to this end they try to find the most extreme databases that still satisfies DP, such as pairs like $[1,1,1,1]$ and $[2,2,0,0]$ (same sum) or $[1,1,1,1]$ and $[2,0,0,0]$. This kind of strategy is not sound (that's not their aim anyway), but as long as the tool is meant as a help for programmers, it's probably a good start.
- For finding a set S given D_0 and D_1 they run the mechanism many times and try to figure out the privacy loss, then selecting the highest ones.
- Given D_0, D_1 and S, they apply a hypothesis test to figure out whether DP is violated: DP being preserved is their null-hypothesis and they try to get a small p-value. I think that makes sense; they have the additional problem that they would need to sample/run the mechanism many times and don't have a ground truth. Instead they use a clever Fisher-test to get a p-value for whether this particular hypothesis is true.

  I like their approach, as it is a nice guide for program designers. The tool runs within 23 seconds, which means it can be used quickly in the development process.

3.3 Secure Computation with Differentially Private Access Patterns

Speaker: Sahar Mazloom (George Mason University), collaboration with S. Dov Gordon (George Mason University)

They look at secure computation; they are looking at secret sharing in a two-party setup. each user performs secret sharing with two servers, but these servers then compute the exact sums in the data. If done naively, we still leak access patterns, in addition to the noiseless results we output. Since generic solutions for hiding access patterns are expensive, they allow a little bit of leakage for these access patterns (i.e., differential privacy). This is related to what Raphael Toledo was working on with his DP Oram project.[4]

The relaxation allows to reduce the asymptotic complexity from $O(n \log n)$ to αn, where α depends on the DP parameters. What they do is that instead of an expensive shuffle operation, they add a number of dummy elements and then perform an oblivious shuffle to mix real and fake elements; each element is annotated with whether it is real or fake (internally, of course).

They can compute a variety of meaningful metrics with their approach including histograms, PageRank and matrix factorization.

3.4 DP-Finder: Finding Differential Privacy Violations by Sampling and Optimization

Speaker: Benjamin Bichsel (ETH Zürich), collaboration with Timon Gehr (ETH Zürich), Dana Drachsler Cohen (ETH Zürich), Petar Tsankov (ETH Zürich), Martin Vechev (ETH Zürich)

[4] Mix-ORAM: Using Delegated Shuffles and

The second system for finding differential privacy violations presented at CCS'18. Their approach introduces a lower-bound for pure differential privacy. They notice that what they actually want to do find some lower bound on the privacy loss (for the whole test S): $\mathcal{L}^S_{M(D_0)||M(D_1)} = \log \frac{M(D_0)\in S}{M(D_1)\in S}$. Their evaluation shows that this is possible, which I don't find extremely surprising.

What's more intesting is that they approximate their privacy losses by sampling, which of course makes sense. They then make their privacy losses differentiable (but they don't seem to order them). They approximate the privacy loss by running $M(D_0)$ many times and $M(D_1)$ many times to then get an approximate for the two probabilities, to then compute the ratio. They do need quite a lot of samples, but I think their sampling approach is pretty interesting.

I think this approach is mostly useful if we're interested in implementation mistakes, not if we care about mechanisms for which we know the probabilities (because then we could look at the real privacy loss).

4 CCS, Wednesday, Cyberphysical Systems session

4.1 Scission: Signal Characteristic-based sender identification and intrusion detection in automotive networks

Speaker: Marcel Kneib, collaboration with Christopher Huth

The talk is about how physical characteristics can be used for identification of engine control units (ECU) in cars.

Motivation Cars can be attacked via hacks, which can be particularly dangerous when human lives are put at risk (failing brakes, hijacked steering, etc.). The speaker presents the issue intuitively in a case where there is no clear separation between critical components and components with connection to the outside world (internet, bluetooth, etc.). An example for that is the controller area network used for vehicular communication. This system doesn't even have sender authentication for the signals that are sent by components. Apparently it is infeasible to introduce proper integrity mechanisms.

Idea and approach Interestingly, the analog CAN signal (how fast does it rise, how stable is it, etc.) can be used to identify ECU's. Scission now samples the signal, slices it, puts them into three possible groups, extracts features and then performs classification.

Their features include the moments of the function, e.g., mean, standard deviation and variance, skewness, etc.; they require their initial training to be in a safe environment (to avoid poisoning attacks). They also share keys between the ECUs and their module. Intrusion detection now means running the classifier to identify the sender; each identifier is only used by one ECU and in case of a conflict, an alarm is raised. To reduce false-positives, they include a confidence bar and only raise an alarm if the classification has a strong confidence in its identification.

Their evaluation considers either 10 or 6+2 ECUs and is tested in real car systems. The identification rate seems pretty good and they have no false-positives. However, I wonder how easy it might be to reduce the confidence in the classification; that appears like a potential vulnerability.

4.2 Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach

Speaker: Hongjun Choi (Purdue University), collaboration with Wen-Chuan Lee (Purdue University), Yousra Aafer (Purdue University), Fan Fei (Purdue University), Zhan Tu (Purdue University), Xiangyu Zhang (Purdue University), Dongyan Xu (Purdue University), Xinyan Deng (Purdue University)

Robotic Vehicles (RV) include hobby drones, delivering drones, military UAC's self driving cars. Abstractly, these RV's are physical systems that interact with the physical world, while under control of a cyber-system. The main motivation for the talk is that while we are starting to understand and defend against cyber attacks, new attacks focus on physical attacks: new signals, spoofed sensors, or throwing

stones at drones. A cute example is disturbing the gyro-sensors using noise (actual audio signals) that will influence the behavior of a drone.

To detect such attacks, they employ control invariants that, e.g., check whether the laws of physics are seemingly violated in the sensor data. To this end, they predict the next system state during runtime and compare this prediction with what they measure. They reverse-engineer the control graph of the RV and insert their invariants into the control program binary of the RV.

They show that their prediction is pretty close to the measurements; their errors are fairly small and at least all of their own attacks were detected. The authors are aware that this doesn't yet make the system secure: They perform an attack and while they immediately detect it in their ground-based system, in the speaker's words, "the drone still crashes, but we see an error message".

4.3 Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems

Speaker: Magnus Almgren (Chalmers University of Technology), collaboration with Wissam Aoudi (Chalmers University of Technology), Mikel Iturbe (Mondragon University)

The talk tackles industrial control systems and their vulnerabilities, e.g., the blackouts caused in Ukraine. Attacks on such systems can be devastating and we need to combine IT with operational technologies to defend against them. ICS systems often are cyclic and deterministic. Thus, "normal" behavious can be learned or even modeled.

Ideas of previous work build a model of the physical process, then use the model to predict future system behavior and compare the predictions with the observations and raise alarms whenever the deviation is too large. Magnus declares that predicting the future is unnecessarily difficult and thus their PASAD system (Process Aware Stealth Attack Detection) focuses on solving an easier problem: They thus require only limited knowledge and also detect subtle and stealthy attacks. PASAD uses the raw features and is model-free.

PASAD works in two phases:

- Offline learning: They extract signals from the system, reduce noise, construct the signal subspace and project training vectors and compute a centroid to define the normal behavior.
- Online detection: They project the same vectors and try to see whether there is a deviation from the "normal" behavior. They raise an alarm whenever the deviation exceeds a threshold.

This is a nice safeguard, but again has the same limitations that the previous talks on this field had: we don't get any kind of guarantee and it is not completely clear, how difficult it would be to fool the system. Still, I think it is a nice monitor that should make attacks more complicated.

4.4 On the Safety of IoT Device Physical Interaction Control

Speaker: Wenbo Ding (Clemson University), collaboration with Hongxin Hu (Clemson University)

Unsurprisingly, smart homes are still on the rise; the amount of devices grows fast and they obtain more and more complicated functions. The talk focuses on physical features and their impact on smart devices, e.g., smart windows might influence the smart heater control. The reverse direction is much more fun: If the attacker can hack into the heater, raise the temperature and thus force the temperature control application to open the window, thus allowing an attacker to open a window and thus break into a house, by hacking into the heater. As another example, an app might check the status of some sensor to lock the house if the owner leaves. Consequently, the app might lock or unlock the physical locks of the house, depending on sensor data.

To protect against such angles we first need to identify all physical angles and then their interactions. To this end, they first analyze their applications for intra-app issues and physical channels. From these two features they build an interaction chain and then they analyze the risk of these chains.

Analysis in more details In their intra-app analysis, they look for flows within the application. They also check which devices might be used and how they can effect each other. Next they try to identify all physical channels from the description using natural language processing. They parse the sentences into words, extract nouns and then infer channels. I'm not quite sure why they don't analyze the sensors directly. They then generate interaction chains to figure out which snippets of interactions can be linked to form larger chains. Finally, they perform a risk analysis; as a base-line they use the intra-app interactions they have previously found (this is why they need them). They assume that intra-app interactions are safe, so they compare their physical interaction chains with the intra-app interaction chains to then classify the former as "probably safe" or "probably not safe". In more details, they actually have a more fine-grained model of the normal behavior: they classify the intra-app chains into several types of chains, e.g., "temperature related stuff" and "movement related stuff"; if the physical interactions fall into the same classes, they are considered fine. If they fall outside of the classes learned on the intra-app interactions, the distance to the nearest class is the "risk factor" of these samples.

Looking at their evaluation, the method seems to leave a lot of room for improvement. It is better than pure random guessing, but not by too much.

5 CCS, Wednesday, Usable Security

5.1 Asking for a Friend: Evaluating Response Biases in Security User Studies

Speaker: Elissa M. Redmiles (University of Maryland), collaboration with Ziyun Zhu (University of Maryland), Sean Kross (University of California San Diego), Dhruv Kuchhal (Maharaja Agrasen Institute of Technology), Tudor Dumitras (University of Maryland), Michelle L. Mazurek (University of Maryland)

Motivation There is an increasing number of surveys published at the top security conferences. A key-question thus is whether the answers given by people questioned for these surveys are correct. There are many aspects, but Elissa focuses on the questions in this talk.

Evaluation / dataset For their evaluation, they use Symantec host records (500k people) on whether and how fast people updated their software and then performed a survey on 2k people how they would intend to respond to a message that there is a new update. They picked the answer choices to match the frequencies they observed in the data. They also phrased the question s.t. it matched previous work. They suggest that you always include an "I don't know / don't want to answer" in any survey, to prevent people from randomly answering if they don't want to answer. They performed a significant number of pretesting steps, but Elissa emphasizes that pilots should only be used for making sure the technical stuff works.

Biases and lessons learned Not surprisingly, they found a systematic bias, i.e., they answer more positively when questioned whether they would update their system. When asking what they would recommend a friend should do instead of what they think they'd do, the answers are even more positive.

They tried to measure the "cost" of updating their system, including "having to restart the system" and "the observed number of crashes" (including observations on the first derivative of crashes: "does the application crash more or less often?"). They found that people who generally tended to update (as by their metric of costs) tended to answer that they intended to update more (significant positive effect); there also was a much smaller effect on whether they indeed updated faster.

To filter out subjects that answered wrong or illogical things, e.g., that claimed they wanted to update because the update didn't require a restart if the message explicitly said a restart was required. Overall though, Elissa paints a bleak picture and suggests not using surveys for finding the actual truth on statistics, but more for getting an impression on "why" people act in certain ways.

5.2 Towards Usable Checksums: Automating the Integrity Verification of Web Downloads for the Masses

Speaker: Kévin Huguenin (UNIL – HEC Lausanne), collaboration with Mauro Cherubini (UNIL – HEC Lausanne), Alexandre Meylan (UNIL – HEC Lausanne), Bertil Chapuis (UNIL – HEC Lausanne), Mathias Humbert (Swiss Data Science Center, ETH Zurich and EPFL), Igor Bilogrevic (Google Inc.)

Motivation Kévin starts by cleverly confronting us with the fact that even we (as security researchers) don't actually check the checksums of software.

Checksums are often put on websites to allow users to verify that software has not been tampered with by an adversary. Since this has to be done manually, Kévin asks the rhetorical question of whether such a barbaric technique is still appropriate in 2018. He asks a few obvious research questions (do people use checksums and are there problems?), gives obvious answers and then improves the state-of-the-art by providing a novel tool for checking integrity.

Surveys:

- In a survey of 2,000 people we see that more than half the people do download software from vendor/developer websites (making them vulnerable to tampering, but also allowing for improvelemt); about a quarter of people remember to have seen checksums. They asked people for the purpose of checksums and find that about 5% or people know that.
- When checking which types of hashes are provided by websites, about half of the ones they looked at used insecure hashes (like MD5) and only a small number of websites included instructions for how to check that the checksum is correct. Finally, they
- They performed a small (n=40) study on how people actually act when confronted with checksums. Most would download software from websites, but only one third was aware of checksums. They then put the subjects in front of an eye-tracker while having them download a file, compute the checksum, check the checksum and then extract annd note down some information about the program, e.g., the version number (a useless instruction used to make sure that participants were not too aware of the aim of the study). The checksum of the third program they downloaded was incorrect, but the beginning and end were correct (which I think is pretty mean): 38% of participants did not detect this mismatch, even though they were explicitly instructed to verify the checksum. This correlation rate was not correlated with prior knowledge about checksums. They also noticed that people mostly looked at the beginning of checksums, not at the end of checksums.

Solution Kévin admits that there already is something like a solution: with SRI (Subresource integrity) the creator of a website can include an integrity field into the `<script>` tag on the page, which will lead to the file not being downloaded if the checksum doesn't match. Consequently, they extended SRI to also work in an `<a>` tag to allow an easier inclusion into download links. Their browser extension extracts the checksums from `<a>`elements, computes the checksums of the downloaded file and displays a message indicating whether the checksums match. Moreover, they provide an extension for Wordpress.

5.3 Investigating System Operators' Perspective on Security Misconfigurations

Speaker: Tobias Fiebig (TU Delft), collaboration with Constanze Dietrich (Berliner Hochschule für Technik), Katharina Krombholz (CISPA Helmholtz Center (i.G.)), Kevin Borgolte (Princeton University)

Motivation Tobias mentions that misconfigurations are a major issue for security. In this work, they started with exploratory interviews using IRC (for some reason) and then performed a study using a questionnaire on about 200 system operators.

Selection of results Over 75% answered that they had made misconfigurations in general, but even 90% admitted that they had made a specific misconfiguration; with the exception of just one operator, everyone answered that they had encountered someone else making a misconfiguration.

They have encountered pretty terrible misconfigurations, including the combination of username `admin` with password `admin` and skipping updates. Things seem to go wrong because of a lack of knowledge, (allegedly not due to poor online resources), overwhelming responsibility (allegedly not due to insufficient funding), and using defaults (allegedly not due to unhelpful standards).

They asked operators on whether they think their managers know what they are doing. Non-IT and governmental OPs were more skeptical of their managers than OPs working in IT. As a funny side-note: trust in their own tools seems to be directly correlated with juniority of the operators, which I think makes sense. Moreover, although OPs like blameless post mortems, i.e., allowing people to honestly report their mistakes and not being punished as long as they were not completely careless, they also answered that their companies did not budget for errors.