Privacy Enhancing Technologies
Lecture 1 – Logistics and Commitment Schemes

Florian Tramèr

AGENDA

1. Course Pitch
2. Course Logistics
3. What is "Privacy"?
4. Rock-paper-scissors over the Phone

1 Course Pitch

$$\underbrace{\text{PRIVACY}}_{\text{I}}\ \underbrace{\text{ENHANCING}}_{\text{II}}\ \underbrace{\text{TECHNOLOGIES}}_{\text{III}}$$

I. This is a course about privacy (in case you were wondering...)

II. We'll mostly look at how to *preserve* privacy (although to motivate why this is necessary we'll sometimes look at attacks on privacy as well).

III. This is a technical class, not a policy class. The class will be fairly mathematical.

This class aims to be both your *first* and *last* class on Privacy Enhancing Technologies (PETS). On one hand, privacy is a topic that is only marginally covered in the CS curriculum, and so this might be your first time learning about topics such as Private Information Retrieval (PIR), Zero-Knowledge (ZK) or Differential Privacy (DP). On the other hand, this class will touch on topics that are at the forefront of current research on PETS. After this class, you will have the tools and background to start reading and dabbling in PETS research.

2 What is "Privacy"?

Glad you asked! According to Wikipedia:

> *"Privacy is the ability of an individual or group to seclude themselves, or information about themselves, and thereby express themselves selectively."*

So privacy is about *control*, and *choice*, of how information is used and shared.

In this (computer science) class, we will focus on how we may *compute* on data in a way that preserves privacy. The general setting we'll consider is that there are multiple parties, each with access to some data x_i, and we want to compute some function

$$y_1, \dots, y_n \leftarrow f(x_1, \dots, x_n)$$

where each party might get a different output y_i, and we want the computation to reveal "not too much" about the input data.

What it means for a function to reveal "too much" about the data will be central to the formal definitions of privacy we'll see in this class.

This generic setting covers pretty much all forms of modern cryptography and privacy-preserving data analysis. For example:

- **Encryption:** You can think of encryption as a two-party protocol where one party (the sender) inputs a message m and the other party (the receiver) inputs nothing. The receiver gets the message m and the sender gets no output.
- **Zero-knowledge:** One party (the prover) wants to prove to the other party (the verifier) that they know some input x such that $f(x) = 1$. Here the verifier inputs nothing and the prover inputs the witness x; the verifier just learns the output $f(x) = 1$ without learning anything more about x.
- **Private retrieval:** A server has a database D and a client wants to retrieve an index i. The computed function outputs D_i to the client and nothing to the server.
- **Private census:** A government wants to release aggregate statistics about the population by collecting survey data x_i from multiple participants, without leaking information about the individual participants.
- **Private machine learning:** Multiple parties have labeled data $(x_1, y_1), \ldots, (x_n, y_n)$, and the function computes the training of a machine learning model M, which is then shared with all parties.
- And many more ...

While the class aims to provide a comprehensive overview of PETS, it is impossible to cover all topics in depth in this class. Here's a non-exhaustive list of topics we *won't* cover:

- Modern two-party computation protocols
- Anonymous and metadata-hiding communication
- Fully homomorphic encryption
- Private payments
- Many differential privacy mechanisms
- ...

A note on *utility*. Generally when deploying privacy-preserving technology, your competition is a *non-private* primitive that has been optimized to death for speed and utility. That is, in many situations, privacy has been sacrificed over time to make way for performance. Winning privacy back without sacrificing (too much) speed is hard.

So when designing PETS we will put a lot of focus on making the primitives fast and simple enough to deploy. There has been a big shift in the PETS research community in this regard over the past few years. Many of the primitives we'll see (especially on the cryptography side) have been known for decades, but were mainly of theoretical interest. But modern variants of these schemes are now deployed (or close to being deployed) in production. In this class, we will often focus on these modern efficient primitives, over more technical constructions that are of historical interest.

3 Course Logistics

Staff:

- Prof. Florian Tramèr
- Jie Zhang, Head TA (PhD student, SPYLab)
- Laura Hetz, (PhD student, Applied Crypto Group)
- Daniel Paleka (PhD student, SPYLab)
- Michael Aerni (PhD student, SPYLab)
- Hana Farid, TA

Website: `https://spylab.ai/teaching/pets-f24/`

Individual questions: florian.tramer@inf.ethz.ch or jie.zhang@inf.ethz.ch

Everything else: Use the Moodle forum for questions about course material, assignments, course policy, etc.

Grading: There will be **four graded homeworks** (worth 25% each). These homeworks will be released on the course website (and announced on Moodle), and are due on Gradescope 2 weeks later (see schedule on course website).

The first homework is out today, and due on Gradescope on October 18th, 5PM.

You must use LaTeX to write your homework. We provide a template on the website. The course code to sign up for Gradescope will be distributed on Moodle.

Collaboration policy: You may discuss the problem sets with other students and you may work together to come up with solutions to the problems. If you do so, you must list the name of your collaborators on the first page of your submission. Each student must write up their problem set independently.

If you use a result from a textbook or research paper in the course of solving a problem, please cite the textbook or paper in your write-up. If you have questions about these policies, please just ask us.

Lectures and exercise sessions: Lecture notes will be posted online for each lecture (typed up notes, and the transcript of the lecture). *The lectures are not recorded.*

Attendance in lectures or exercise sessions is not mandatory, but is strongly recommended. Since there are no exams in this class, the problem sets are supposed to be challenging! So start early and ask questions in the exercise sessions.

Prerequisites: This is not an "introductory" course. While we strive to introduce all the necessary background in class, some experience with complexity classes, cryptography, and probabilities will be helpful. We provide a background sheet to refresh these topics.

Feedback: This is the first time we teach this class, so we would love to get feedback on how to improve it! You can use this form to give us anonymous feedback throughout the semester. You can fill it out as often as you want. And we make mistakes! If something looks wrong or impossible, please let us know.

4 A Cool Application: Rock-paper-scissors over the Phone

We'll start off with a very simple form of privacy primitive, which will also play a key role in more advanced crypto schemes we'll cover later in the course. It is also a nice illustration that privacy techniques can have applications in protocol design that go beyond "protecting personal data".[1]

Suppose that you and your friend are on the phone and you need to choose where to have dinner. You want Chinese and they want Italian, and so you need a tie breaker. You decide to play a game of rock-paper-scissors.[2]

If you were next to each other, you could just play a round of rock-paper-scissors and make sure there is no cheating. But how would we do this over the phone? Whenever you start announcing "rock..." your friend might quickly change their mind, announce "paper", and claim there was a slight delay on the line.

So we need a way to "synchronize" the two announcements, so that one party cannot choose their answer based on the other party's answer. How could we do this? Let's consider two natural cryptographic solutions that don't quite work:

- *Hashing:* You pick your choice $x \leftarrow \{0,1,2\}$ and send a cryptographic hash $h \leftarrow H(x)$ to your friend. Your friend picks their own choice $x' \leftarrow \{0,1,2\}$ and sends it to you in the clear. You then reveal x and your friend checks that $h = H(x)$.

  Unfortunately, you won't get your Chinese food! Your friend can just check if $h = H(0)$,$h = H(1)$ or $h = H(2)$ and then set x' accordingly to win. The problem is that the hash doesn't *hide* your choice x.

- *Encryption:* So let's use a stronger form of data hiding, encryption. You pick a key k for a semantically-secure encryption scheme and you send $c = \texttt{Enc}(k, x)$ to your friend, who sends you x' as before. You then reveal (k, x) and your friend checks that $\texttt{Dec}(k, c) = x$.

  The attack above no longer works. Because of the encryption scheme's randomness, your friend cannot tell if you encrypted a $0, 1$ or 2. The problem is that now *you* can cheat! Specifically, you might be able to find a key k' such that $\texttt{Dec}(k', c) \neq x$. For example, suppose you used the most secure encryption scheme of all, the one-time pad. So you sent $c = x + k \pmod 3$ to your friend, where k is a random value in $\{0,1,2\}$. But then, after seeing x', you can decide to send any other value $(x + \Delta, k - \Delta)$ to your friend. Chinese always wins! The issue here is that most encryption schemes are not *binding*: you can selectively choose a key to decrypt to any message.

To resolve this issue, we will need a primitive that is both *hiding* and *binding*. This is called a *commitment scheme* [Blu83]. You can think of a commitment scheme as a box that you can lock with a key. Once you lock your answer in the box, you can give it to your friend, and thereafter you cannot change your answer and your friend cannot learn anything about it. Later, you can give your friend the key to reveal your answer.

4.1 Commitment Schemes

Before we define commitment schemes formally, we briefly have to talk about the notion of *indistinguishability*.

[1] We'll see more examples of this when we talk about Differential Privacy in the second half of the course.

[2] The original application proposed by Blum [Blu83] was coin flipping over the phone. But an actual fair coin flip is a tiny bit more complicated than in practice, since you need to guard against biased coins. So the actual protocol ends up essentially being a form of rock-paper-scissors.

Indistinguishability. Very often in a PETS, the adversary will observe the output of some randomized process applied to data, i.e., $f(x,r)$ where $r \in \mathcal{R}$ is randomly chosen. We can then define the distribution $\{f(x,r) : r \xleftarrow{\$} \mathcal{R}\}$ of the data seen by the adversary (which is often called the adversary's *view*). To argue privacy, we then show that the adversary's views for different data x, x' are indistinguishable. This indistinguishability can be either *statistical* or *computational*:

Information-theoretic (statistical):

$$\{f(x,r) : r \xleftarrow{\$} \mathcal{R}\} \equiv \{f(x',r) : r \xleftarrow{\$} \mathcal{R}\}$$

In words, the distributions of the views are identical (or negligibly close).

Computational: For all *probabilistic poly-time* (PPT) adversaries $\mathcal{A}$, we have

$$\left| \Pr\left[\mathcal{A}(y) = 1 \mid y \leftarrow f(x,r)\right] - \Pr\left[\mathcal{A}(y) = 1 \mid y \leftarrow f(x',r)\right] \right| \leq \mathsf{negl}(\lambda) \,.$$ [a]

That is, the adversary cannot guess with non-negligible advantage whether the function was applied to x or x'. Note that in the equation above, the probability is taken over the randomness r of the function and the (potential) randomness of the adversary.

More succinctly, we say that the distributions $\mathcal{D}_1 := \{f(x,r) : r \xleftarrow{\$} \mathcal{R}\}$ and $\mathcal{D}_2 := \{f(x',r) : r \xleftarrow{\$} \mathcal{R}\}$ are *computationally indistinguishable*, denoted as $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$.

The parameter λ is the *security parameter* which controls the strength of the cryptographic assumptions underlying the indistinguishability (e.g., the size of a key for an encryption scheme, or the size of a group in a discrete logarithm setting). We often leave this parameter implicit, but technically the function f and the ranges for x and r are all parameterized by λ.

[a] A function $f(n)$ is negligible if $f(n) = o(n^{-c})$ for all constants $c \in \mathbb{N}$.

A formal definition of commitment schemes.

Definition 1 (Commitment scheme). A commitment scheme is an algorithm $\texttt{Commit} : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ that takes in a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ and outputs a commitment $c \in \mathcal{C}$ such that $\texttt{Commit}(m,r) = c$.

Such a scheme should satisfy two properties:

(Statistical) Hiding: the commitment c should leak nothing about the message m:

$$\forall m_0, m_1 \in \mathcal{M}, \{\texttt{Commit}(m_0,r) : r \xleftarrow{\$} \mathcal{R}\} \equiv \{\texttt{Commit}(m_1,r) : r \xleftarrow{\$} \mathcal{R}\}$$

(Computational) Binding: after sending the commitment c, the adversary cannot "change their mind": No PPT adversary $\mathcal{A}$ can produce $m_0, m_1 \in \mathcal{M}$ and $r_0, r_1 \in \mathcal{R}$ such that

$$\texttt{Commit}(m_0,r_0) = \texttt{Commit}(m_1,r_1) \text{ and } m_0 \neq m_1 \,.$$

We can also define commitment schemes where the hiding property is computational and the binding property is statistical. But we cannot get both properties to be statistically secure at the same time (at least not in the "standard" model, more on this later).

A note on terminology: while we define commitment schemes here as having a single algorithm, `Commit`, in practice commitment schemes are used as a two-stage protocol:

1. One party sends a *commitment* c to a message m to the other party.
2. The same party later sends a *opening* r to the commitment. The other party then checks that $\texttt{Commit}(m, r) = c$.

4.2 The Simplest Commitment Scheme in the World

Remember our failed attempt at coin flipping over the phone, where you sent a hash $H(b)$ to your friend? We'll here's what you should have done:

$$\texttt{Commit}(m, r) = H(m, r) .$$

That's it! So simple...

How do we prove this is secure? Unfortunately, "standard" security properties of hash functions (e.g., collision resistance) are not quite enough.[3] To prove security of this scheme, we will need

The Random Oracle Model

4.3 The Random Oracle Model

Many (most) cryptographic schemes that are used in practice rely on a heuristic: these schemes make use of a hash function (e.g., like SHA-3) which is very complicated to analyze. And so when proving security, we assume (pretend) that this hash function is actually a *completely random function* $H : \mathcal{X} \to \mathcal{Y}$, where for each $x \in \mathcal{X}$, the output $H(x)$ is sampled uniformly at random from $\mathcal{Y}$ [BR93].

Of course, this is not true in practice. A truly random function would require exponential space to define. But this trick is super useful to prove security. Let's see an example for our previous commitment scheme:

$$\texttt{Commit}(m, r) = H(m, r) .$$

Hiding: If H is a random function, then $H(m, r)$ is uniformly random over $\mathcal{C}$ as long as the adversary doesn't evaluate H on (m, r). If r is large enough (e.g., 128 bits), this happens with statistically negligible probability.

Binding: To break binding, the adversary has to find $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$ such that $H(m, r) = H(m', r')$. But a random function is collision resistant, so this is hard.

[3]To provide some intuition, consider a dummy cryptographic hash function that always appends (part of) the message m to the end of the hash. This still satisfies all the properties we want from a hash function (e.g., one-wayness, collision resistance), but it is definitely not hiding.

Controversy: The Random Oracle Model is somewhat controversial. The reason is that we know (for sure) that any hash function we'll use in practice is most definitely *not* a random function. This is also why we call this a model and not an assumption (such as "Factoring is hard"). The latter is possibly true (and indeed that's what we assume), while we know for certain that the first is false (but it's a convenient model to work in).

We can build many very efficient and simple schemes that are secure in this model. The heuristic is then to replace the random function by a *suitable hash function* when deploying the scheme. To break the scheme, the adversary would then have to exploit some structure or property of the hash function that distinguishes it from a random function. Since we don't know of any such properties for cryptographic hash functions such as SHA-3, we believe the scheme remains secure.

Some (more theoretically inclined) cryptographers are not quite happy with this heuristic, and prefer to build schemes that can be proven secure under specific hardness assumptions. Unfortunately, these schemes are then often (much) more complicated and harder to deploy. It is also known that in some (highly contrived) cases, schemes that are proven secure in the random oracle model are actually *insecure* when instantiated with *any* concrete hash function [CGH04].

4.4 Pedersen Commitments

Let's now see a commitment scheme where we don't need to assume a random oracle model. This scheme [Ped91] will be less efficient than the previous one, although still quite practical. The main reason it is interesting is because it has other nice properties.

Setup: Let $\mathbb{G}$ be a group of prime order p, and let $g, h \in \mathbb{G}$ such that the discrete log between g and h (i.e., $g^x = h$) is unknown.

Commitment: $\texttt{Commit}(m, r) := g^m h^r$, for $m, r \in \mathbb{Z}_p$.

Why is this scheme cool? It is *linearly homomorphic*, i.e., you can combine commitments to different messages and get a commitment to the sum:

$$\texttt{Commit}(m_0, r_0) \cdot \texttt{Commit}(m_1, r_1) = \texttt{Commit}(m_0 + m_1, r_0 + r_1) .$$

We can thus *compute* (linear functions) over hidden data!

Security. Let's analyze the security of this scheme:

Hiding: The scheme is perfectly hiding. We have:

$$\texttt{Commit}(m, r) = g^m h^r = g^m \cdot g^{x \cdot r} = g^{m + x \cdot r} .$$

Given any commitment $c = g^\alpha$, and any message m, there exists exactly one value of r such that $g^m h^r = c$, namely $r = (\alpha - m)/x$. Since r is uniformly random, c is equally likely to be a commitment to any message m.

Binding: The scheme is computationally binding, assuming the hardness of the *discrete logarithm problem* in $\mathbb{G}$.

Discrete Logarithm Problem (DLP): Given a group $\mathbb{G}$ with generator g, and an element $h \in \mathbb{G}$, it is hard to find x such that $g^x = h$.

Let's see how to prove this. We want to show a *hardness reduction:* an attacker $\mathcal{A}$ that can break the binding property can be used to find discrete logarithms.

Life advice: If you want to break a discrete log, try to find two different representations of some group element c:

$$\begin{aligned} g^{m_0}h^{r_0} &= c = g^{m_1}h^{r_1}, \qquad m_0 \neq m_1 \\ g^{m_0}(g^x)^{r_0} &= g^{m_1}(g^x)^{r_1} \\ m_0 + x \cdot r_0 &= m_1 + x \cdot r_1 \quad \Longrightarrow \quad \boxed{x = \frac{m_1 - m_0}{r_0 - r_1}} \end{aligned}$$

Given an instance (G, g, h) of the discrete logarithm problem, we can construct parameters (G, g, h) for the Pedersen commitment scheme. We then pass these to our binding adversary $\mathcal{A}$, who returns

$$m_0, m_1, r_0, r_1$$

If the binding attacker $\mathcal{A}$ succeeded, then it holds that $g^{m_0}h^{r_0} = g^{m_1}h^{r_1}$. Then we can apply the life-advice above to find x. So we can recover the discrete log with the same probability as our attacker $\mathcal{A}$ breaks the binding property. (and so if we assume that the discrete logarithm problem is hard to solve for any PPT adversary, the scheme is computationally binding).

References

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. *Cryptology ePrint Archive*, 2015.

Why Care About Privacy Enhancing Technologies?

What would the world look like if we couldn't expect any privacy for some of our communications or interactions. The world (probably) wouldn't end, but it would not be a very nice place either. As beautifully put by Philip Rogaway [Rog15]:

"Cryptography rearranges power: it configures who can do what, from what."

And yet, our privacy is under constant threat.

Many governments are not happy with the idea of encryption they cannot break, and push for legislation to ban (or backdoor) end-to-end encryption.[4] You might say *"but I have nothing to hide!"*, and sure, you're not a terrorist or criminal. But the government may one day criminalize your decision to have an abortion[5], or to protest that government...[6]

And encryption is just the tip of the iceberg. Even if your communications are encrypted, your online activity and communication patterns are still tracked wherever you go. Social media apps might know so much about you that they inadvertently (or on purpose) leak your health status[7] your sexual relationships,[8] or the location of your army base![9]

Such metadata has been used by governments and companies to target people when they cannot read their (encrypted) messages, sometimes to fight terrorism[10], but also to target people based on their sexual orientation[11], or (again...) for their health choices.[12]

The rapid progress of AI doesn't help either. The data you post online now powers large-scale facial recognition systems[13] and fuels deepfake crimes ranging from non-consensual pornography[14] to phone or video scams targeting your friends and family.[15]

When companies "try" to protect data, they also often get it wrong. The field of data "anonymization" is littered with examples of data releases that were not. E.g., data of taxi cab rides in New York City which leaked drivers' incomes and home addresses,[16] or the re-identification of Netflix customers from released movie ratings.[17]

Sounds bleak? Well this class will try to show what we* can *do (on the technological side at least) to protect our privacy while still doing cool computery things.

[4] https://www.forbes.com/sites/digital-assets/2024/05/07/european-threat-to-end-to-end-encryption-would-invade-phones/

[5] https://www.washingtonpost.com/technology/2022/07/03/abortion-data-privacy-prosecution/

[6] https://cybernews.com/news/how-encrypted-messaging-changed-the-way-we-protest/

[7] https://splinternews.com/facebook-recommended-that-this-psychiatrists-patients-f-1793861472

[8] https://www.cbsnews.com/sanfrancisco/news/uber-crunches-user-data-to-determine-where-the-most-one-night-stands-come-from/

[9] https://www.wired.com/story/strava-heat-map-military-bases-fitness-trackers-privacy/

[10] https://abcnews.go.com/blogs/headlines/2014/05/ex-nsa-chief-we-kill-people-based-on-metadata

[11] https://www.politico.com/news/2024/02/13/planned-parenthood-location-track-abortion-ads-00141172

[12] https://www.vox.com/recode/22587248/grindr-app-location-data-outed-priest-jeffrey-burrill-pillar-data-harvesting

[13] https://www.nytimes.com/2020/01/18/technology/clearview-privacy-facial-recognition.html

[14] https://www.techtimes.com/articles/301757/20240218/deepfake-dangers-rise-ai-generated-pornography-sparks-global-concerns.htm

[15] https://edition.cnn.com/2024/02/04/asia/deepfake-cfo-scam-hong-kong-intl-hnk/index.html

[16] https://www.theguardian.com/technology/2014/jun/27/new-york-taxi-details-anonymised-data-researchers-warn

[17] https://www.wired.com/2007/12/why-anonymous-data-sometimes-isnt/