Lecture 1

Henry Corrigan-Gibbs
CS355 - Spring 2019
Lecture 1

CS355 - Lecture 1

April 1, 2019

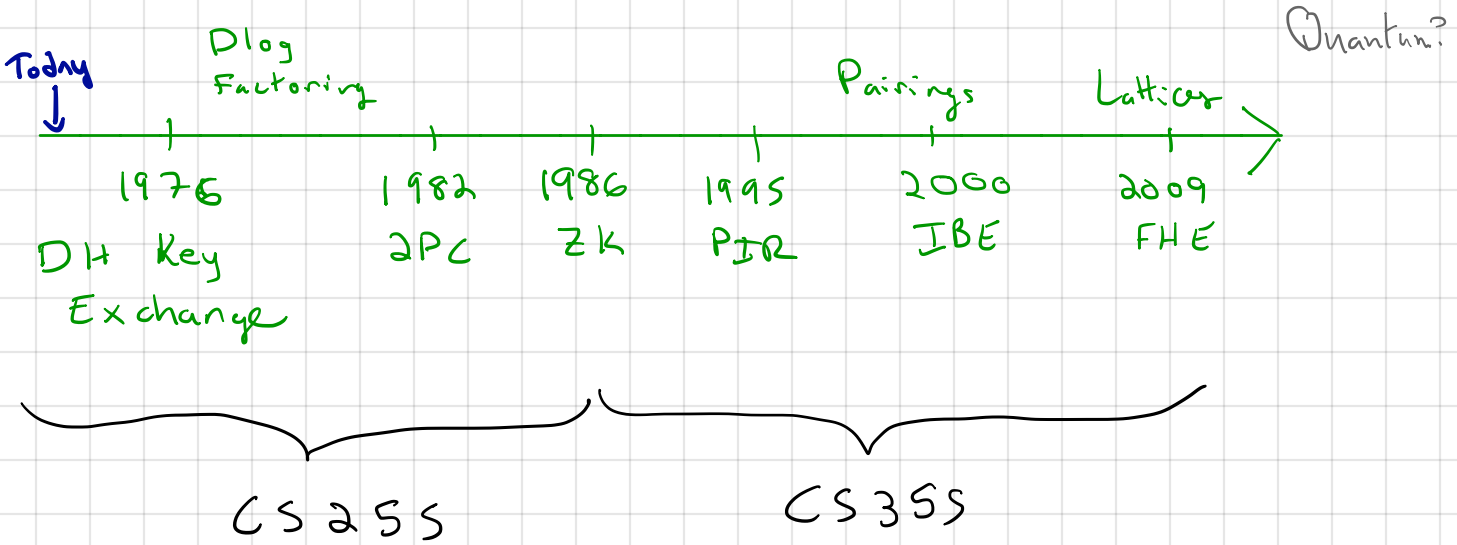
Instructors: Dima Kogan
Florian Tramèr
Henry Corrigan-Gibbs
} PhD students w/ Dan Boneh

TA: Katy Woo

Plan

- Goals of the course
- Logistics
- Merkle's key exchange scheme
  - * Protocol
  - * Random-oracle model
  - * Security analysis

Goals of this Course



Questions we will answer:

* How can you read a row from a database without the database learning which row you read?
* If I have an input x (e.g. my email inbox) and you have a program f (e.g. a spam filter) how can I learn $f(x)$ without:
  - you learning x or
  - me learning f?
* How do you factor big integers?
* How can you compute arbitrary functions on encrypted data? ("fully homomorphic enc.")
* How do you build cryptosystems that even quantum computers can't break?

... and much more!

Our goals

1. Your first advanced course in cryptography.
   * Important formalisms
   * Prepare you to do research in crypto
   * Expose you to open research questions
   ↳ Last year's students in PhD programs!

2. Your last advanced course in cryptography.
   * Give you a taste of cutting-edge crypto tools
   * Allow you to read & understand crypto papers
   * Prepare you to use crypto to change the world!
   ↳ Last year's students at tech companies doing crypto implementation!

Above all: This course should be fun!

Our goal is to show you how beautiful and surprising and powerful modern cryptography can be.

We love this stuff and we hope that by the end of the course, you will too...

Logistics

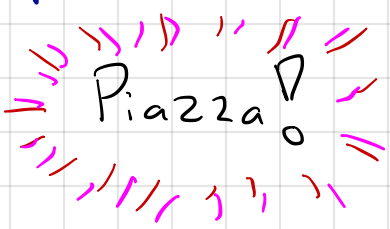
← The spinach

Website: https://cs355.stanford.edu

Instructors: cs355@cs.stanford.edu

↑ For individual questions

For all questions about course material, problem sets, course policies, ...



Office hours are on the website.

↳ If you want to talk with us 1-on-1 (e.g. if you're struggling in this class or with another class, if you want to ask advice about school- or non-school issues), email us.

Students are priority #1, so we will try to make time to meet with you.

Problem Sets

* 5 total, one every 2 weeks
* First due **Friday, April 12 @ 5pm via Gradescope.**

↑ See course website for details. HW online now.

* You must use Latex!
  It's an excellent tool

Logistics

Lecture Notes On website soon after class.
↳ But please attend lecture ☺

Feedback There is an anonymous feedback form on the website. Please use it to send feedback to us throughout the quarter.

Note: CS 355 = CS 255 + 100

Since there are no exams in this course, the problem sets are supposed to be challenging.

⇒ Start early!

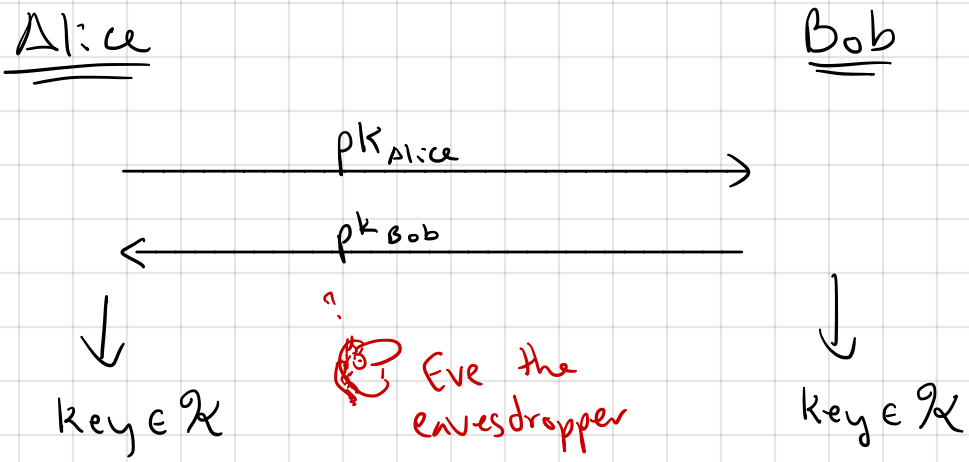
Come to OHs!

Also: We make mistakes.

If something looks incorrect or impossible, please let us know via Piazza.

Merkle Puzzles & Key Exchange

"Key exchange" is the problem that launched public key cryptography.



Alice and Bob exchange some messages over a public channel,

→ They agree on a shared secret key.
→ A passive attacker that sees all messages "learns nothing" about key.

(Q: What commercial app made this problem important?)

Q: What properties do we need of key ex.?

Two goals: 1. Correctness. Agree on same key.

2. Security. No "efficient" attacker can distinguish true key from random. ↑ What does eff. mean formally?

In CS255 you saw how to build key exchange from.....

- DH problem
- RSA problem
- .... any public-key encryption system.

In these systems, Alice and Bob run in poly time, best attack runs in super-poly time.

...But these ideas didn't exist until 1976.

Merkle Puzzles (1974)

- Predated D-H key exchange (1976)
- Uses only hash functions (no crazy PK crypto)
- Conceived by Ralph Merkle as a project for his undergrad comp. security class (!!!)
- The catch: Alice & Bob run in time $\approx O(1)$
  Best attack runs in time $\approx O(1^2)$
  Quadratic gap, not exponential

Still, its a beautiful idea---so simple.

...also a reminder that you need no formal background to have great world-changing ideas.

... See HW for major research questions on this.

Key Background

In this course, an "efficient" algorithm is a randomized one that runs in polynomial time.

↑ Q: Polynomial time in what???

A: In the "security parameter" ... often called λ.

Think of λ as the bit length of the secret keys ... $\lambda \approx 128$, 256.

Idea: In all systems we design (except today)

* Honest algs run in time $\text{poly}(\lambda)$.
* Best attack runs in time super-poly in λ.

[e.g. For elliptic-curve DH key exchange

Alice and Bob: $\approx \lambda^3$ time

Best attack $\approx 2^{\lambda/2}$ time]

↑ Secure against all attackers running in $\text{poly}(\lambda)$ time.

Formally,

Non-Interactive Key Exchange

$1^\lambda = 1111\cdots1$ (λ times)

Why do we do this?

Three efficient algorithms

$\text{Setup}(1^\lambda) \to pp$ Outputs public parameters

$\text{Publish}(pp) \to (sk, pk)$ Outputs public part, secret part

$\text{KeyGen}(sk, pk) \to \text{Key} \in \mathcal{K}$ Outputs shared secret

Merkle's Key Exchange

Uses a hash fn $H: \underbrace{\mathbb{Z}_{\lambda^2}}_{\text{Ints mod } \lambda^2} \to \{0,1\}^\ell$ [e.g. $\ell = \ell_y \lambda^2$]

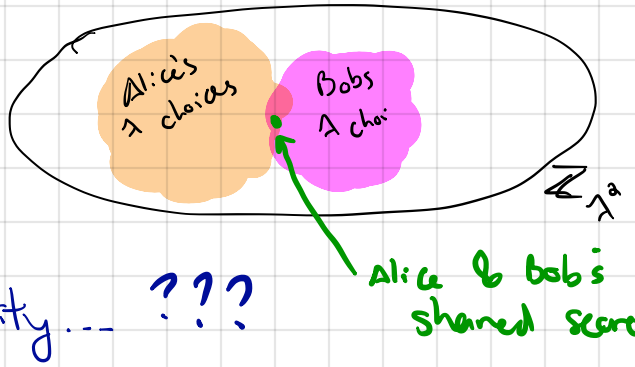
Alice	Bob
1 Pick ints $a_1, \dots, a_\lambda \xleftarrow{R} \mathbb{Z}_{\lambda^2}$	
	$\xrightarrow{H(a_1), \dots, H(a_\lambda)}$
	2 Picks ints $b_1, \dots, b_\lambda \xleftarrow{R} \mathbb{Z}_{\lambda^2}$
	$\xleftarrow{H(b_1), \dots, H(b_\lambda)}$
3 Find least i,j s.t. $H(a_i) = H(b_j)$	
Output a_i as shared secret.	Do same as Alice - Output b_j as shared secret.

Sanity Checks

1: Efficiency... Both Alice and Bob run in time $\tilde{O}(\lambda)$

$\tilde{O}(\cdot)$ hides $\log \lambda$ factors.

2: Correctness... on your HW.



3: Security... ???

Note: There is a time-$O(\lambda^2)$ attack!

Security Properties

What properties do we need?

1. Correctness. $\forall$ $pp \leftarrow Setup(1^\lambda)$

$(sk_A, pk_A) \leftarrow Publish(pp)$

$(sk_B, pk_B) \leftarrow Publish(pp)$

$$Pr\left[KeyGen(sk_A, pk_B) = KeyGen(sk_B, pk_A)\right] \geq 1 - negl(\lambda)$$

↑ Chance that Alice & Bob disagree is very small

A "negligible function" $f(\lambda)$ is one such that

$f(\lambda)$ is $O(\frac{1}{n^c})$ for all constants $c \in \mathbb{N}$.

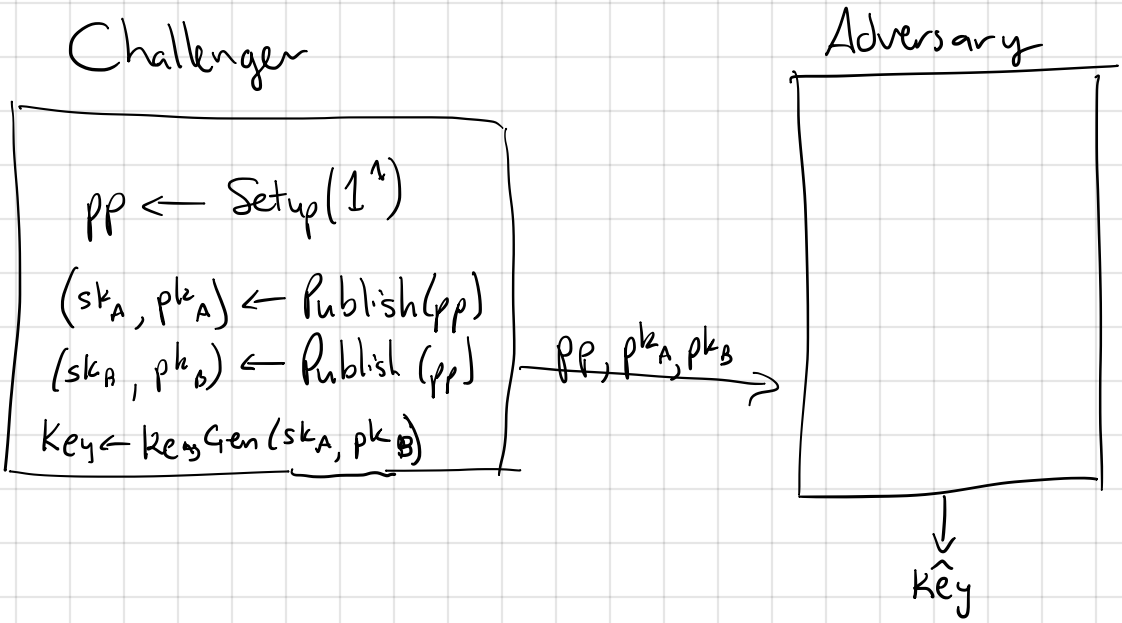
(Or, $f(\cdot)$ is negligible if its inverse grows faster than any fixed polynomial.

e.g. $2^{-\lambda}$, $2^{-\sqrt{\lambda}}$, $\lambda^{-\log\lambda}$, $\lambda^{\log\log\log\lambda}$, ...

Q: Why is this the right defn of "negligible"?

2. Security... Defined by a game.

We're going to prove that Merkle's protocol almost satisfies the following (weak) notion of security.



We say that key exchange is "weakly secure" if, for all efficient adversaries, in the above experiment,

$$\Pr[Key = \hat{Key}] \leq negl(\lambda).$$

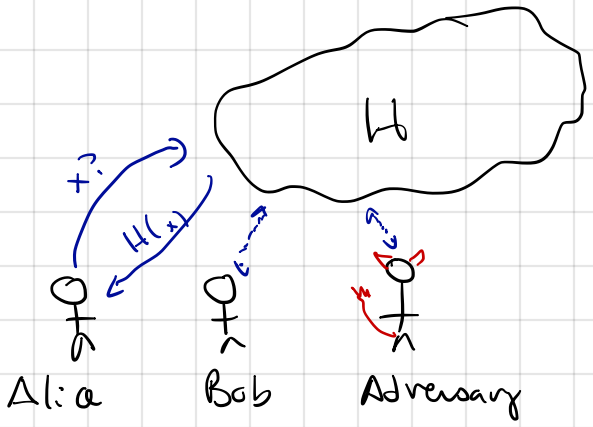
Why is this notion weaker than ideal?

[For Merkle's scheme, we only prove security against advs running in time $T = o(\lambda^2)$.]

Sketch of Security Analysis

We will use the "random-oracle model," introduced to crypto by Bellare and Rogaway (1993)

Idea: Think of hash fn H as a truly random function to which all parties have oracle access



* R.O. Model makes security analysis vastly simpler in many cases.

* In practice, replace random oracle w/ SHA256 and hope that nothing breaks.
  ↳ It works shockingly well.

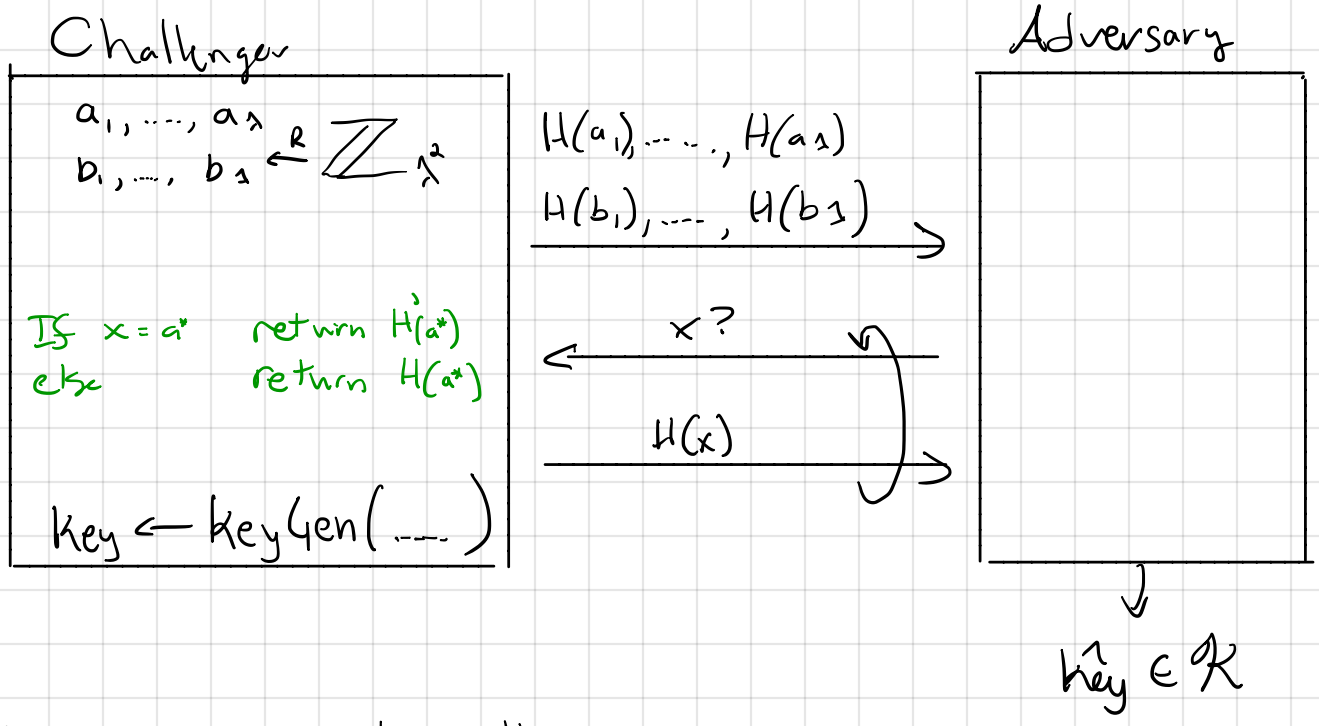
Controversial!

1) Where can you get a random oracle? SHA256 isn't a random fn!

2) Unsafe in general:
   ∃ sig schemes that are secure in R.O. model but ∀ eff hash fns H are insecure when instantiated using H. (Canetti, Goldreich, Halevi 1998)
   ↳ These counter examples are somewhat contrived

Security Analysis

First, let's show that if Adv runs in time $o(\lambda^2)$, prob that adv recovers key is $< 1/2$.



We want to show that

$$W_0 = Pr[Key = \hat{key} \text{ in } Exp0] \leq \text{small}$$

↖ Prob that adv wins in Experiment 0

Idea: Modify experiment.
→ Challenger responds to R.O. queries using H' instead of H on special point a^* ← secret key.
→ In this world, key is indep of adversary's view... random guessing is best can do.

$$W_1 = Pr[Key = \hat{key} \text{ in } Exp\,1] \leq \frac{1}{\lambda^2}$$

Security Analysis

Need now to only show that adv's behavior in Exp 0 and Exp 1 are "similar."

Let F denote the event that Adv queries H on point a^*

Let T denote the max # of queries that Adv makes.

Claim $\Pr[F] \leq \frac{T}{\lambda^2}$.

Pf idea * The value a^* is independent of A's view
* On the ith query, $\Pr[A \text{ queries } a^*] \leq \frac{1}{\lambda^2}$.

By union bound $\Pr[F] \leq T \cdot \left(\frac{1}{\lambda^2}\right)$. ■

To complete proof, see that

$$W_0 \wedge \overline{F} \text{ occurs} \iff W_1 \wedge \overline{F}$$

Then by Difference Lemma (Bohet-Shoup, Thm 4.7)

$$|\Pr[W_0] - \Pr[W_1]| \leq \Pr[F].$$

$$\Rightarrow \Pr[W_0] \leq \Pr[W_1] + \Pr[F] \leq \frac{T+1}{\lambda^2}.$$

⟹ We want a protocol in which adv running in time $T = o(1)$ has negligible probability of recovering key.

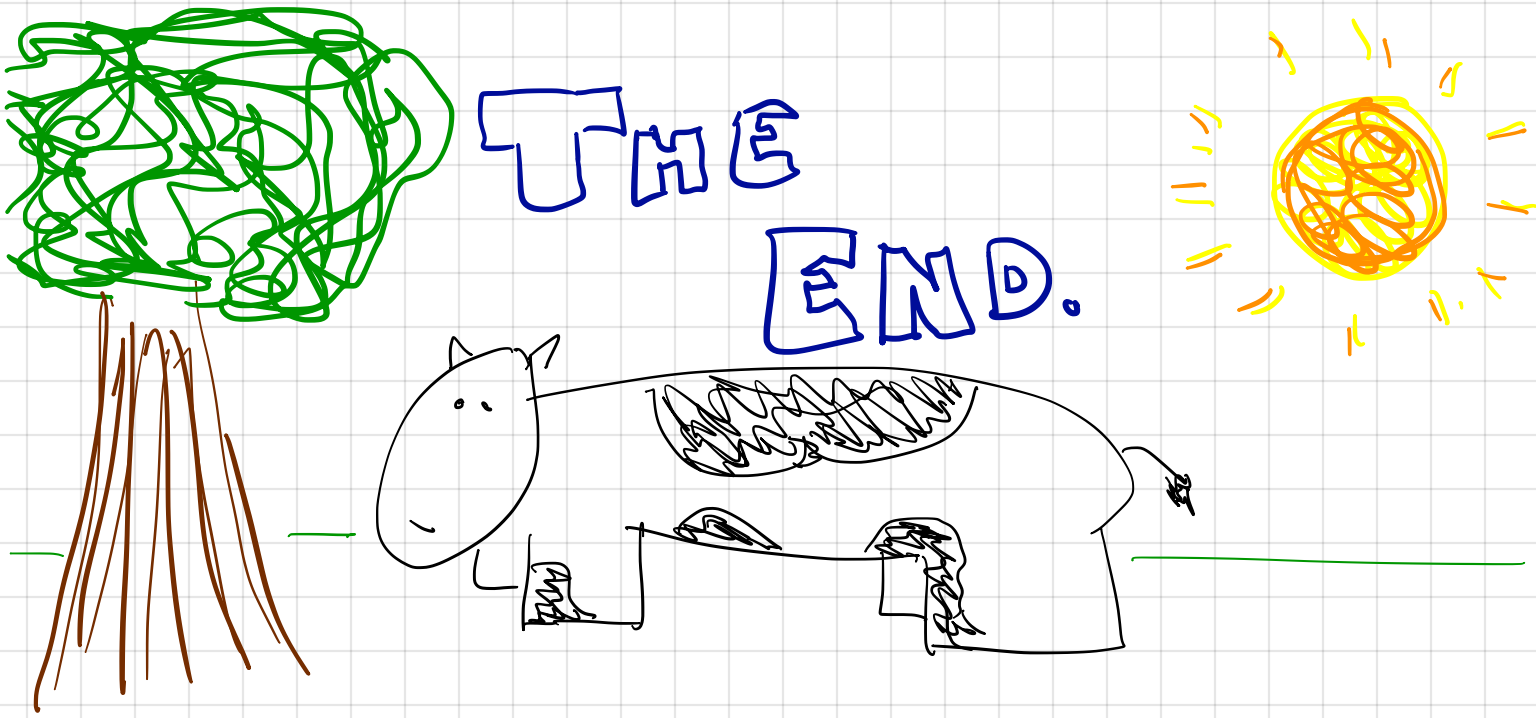
Idea: * Run protocol $k = \log^2 \lambda$ times in parallel.

* Alice and Bob XOR together the keys they get from the k rounds.

* Idea: Adv has to win k times to break protocol

$$\Pr[\text{Adv wins } k \text{ times}] \leq \left(\frac{1}{2}\right)^k$$

$$\leq \left(\frac{1}{2}\right)^{\log^2 \lambda}$$

$$\leq \left(\frac{1}{\lambda}\right)^{\log \lambda}$$

$$\leq \text{negl}(\lambda).$$