Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 11

Portland State University
Lecturer: Fang Song

Feb. 13, 2018

Draft note. Version: February 20, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) CCA/AE formal definitions, foundations
- Computational indist. and hybrid argument
- Public key revolution; review of number theory
- Review of HW2/Quiz2

Computational indistinguishability

We have talked about "indistinguishability" quite a bit so far without a formal treatment. For example, we say the encryptions of m_0 and m_1 are hard to distinguish in a computationally secret encrytion; a pseudorandom string (i.e. output from a PRG) is hard to distinguish from a truly random string. Note that "hard" only holds against computationally bounded (i.e. PPT) adversaries. It's time to bring up the formal notion of *computational indist.*. To do so (in an asymptotic approach), we need to talk about *probability ensembles*.

We will only be concerned with probabliity ensembles indexed by natural numbers. For instance consider $\mathcal{X} := \{X_n : n \in \mathbb{N}\}$, where X_n denotes a distribution for each n (e.g., uniform over $\{0,1\}^n$). $\mathcal{X}$ needs to be *efficiently samplable*[1]. In cryptosystems and games, each security parameter n induces various distributions we care about (e.g., $E_k(0)$ where $k \leftarrow G(1^n)$), and the collection of them for all n naturally gives a probability ensemble.

[1] There is a PPT algorithm S such that $S(1^n)$ and X_n are identically distributed.

Definition 1. Two probability ensembles $\mathcal{X} = \{X_n : n \in \mathbb{N}\}$ and $\mathcal{Y} = \{Y_n : n \in \mathbb{N}\}$ are computationally indistinguishable, denote $\mathcal{X} \approx_c \mathcal{Y}$, if for every PPT disginguisher D

$$\mathbf{Adv}_D^{\mathcal{X},\mathcal{Y}} := \left| \Pr_{x \leftarrow X_n}[D(1^n, x) = 1] - \Pr_{y \leftarrow Y_n}[D(1^n, y) = 1] \right| \leq \mathrm{negl}(n)\,.$$

As an example we can restate the condition of a PRG $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ as

$$\{G(U_n)\} \approx_c \{U_{\ell(n)}\},$$

where U_m denotes uniform distribution on $\{0,1\}^m$.

We will often just say that two distributions (random variables) for a particular n are computationally indistinguishable, without refering to the ensembles explicitly.

You may be wondering, is there a notion of *non-computational* indist.? Yes. There is *statiscal* indist., which we denote $\mathcal{X} \approx_s \mathcal{Y}$. We just remove the PPT restriction in definition 1. It is equivalently characterized to a distance mesure *statistical distance*. It is sometimes helpful (but not always correct) to think of $\mathbf{Adv}^{\mathcal{X},\mathcal{Y}}$ as a computational analogue of distance measure of two distributions. If $\mathcal{X} = \mathcal{Y}$, then they are *perfectly* indistinguishable. Do you recall any such examples?

Lemma 2. *$\approx_c$ is "bounded-transitive". Namely if $\mathcal{X} \approx_c \mathcal{Y}$ and $\mathcal{Y} \approx_c \mathcal{Z}$, then $\mathcal{X} \approx_c \mathcal{Z}$. This holds up to any polynomially many applications.*

Hybrid argument

We use the parallel composition of a PRG to illustrate a very intuitive yet powerful technique to prove security.

Given $G : \{0,1\}^n \to \{0,1\}^{n+1}$, construct $PG : \{0,1\}^{tn} \to \{0,1\}^{t\cdot(n+1)}$

$$PG(s_1, \ldots, s_t) := G(s_1) \| \ldots \| G(s_t).$$

Theorem 3. *PG is a PRG.*

Proof. We want to show $PG(U_{tn}) \approx_c U_{t(n+1)}$. Define the following *hybrid* distributions:

$$\begin{aligned}
Z_0 := PG(U_{tn}) &= G(s_1)\|G(s_2)\| \ldots \|G(s_t), \quad s_i \leftarrow U_n\,; \\
Z_1 &:= G(s_1)\| \ldots \|G(s_{t-1})\|U_{n+1}\,; \\
&\vdots \\
Z_{t-1} &:= G(s_1)\|U_{n+1}\| \ldots \|U_{n+1}\,; \\
Z_t := U_{t(n+1)} &= U_{n+1}\| \ldots \|U_{n+1}\,.
\end{aligned}$$

We claim that $Z_i \approx_c Z_{i+1}$ for all $i = 0, \ldots t-1$, which will imply $Z_0 \approx_c Z_t$. Suppose for contradiction that $Z_i \not\approx_c Z_{i+1}$.

$$\left| \Pr_{z \leftarrow Z_i}[D(z) = 1] - \Pr_{z \leftarrow Z_{i+1}}[D(z) = 1] \right| \geq 1/\operatorname{poly}(n).$$

Then we construct D' to distinguish $G(U_n)$ from U_{n+1}.[2]

[2] Draw reduction diagram

D' is given a string r, either pseodurandom or truly random

1. D' samples uniformly random seeds $s_1, \ldots, s_i \leftarrow \{0,1\}^n$ and generates $G(s_1) \ldots G(s_i)$; D' also samples $r_{i+2}, \ldots r_t \leftarrow \{0,1\}^{n+1}$ at random.

2. Let $z := G(s_1)\| \ldots \|G(s_i)\|r\|r_{i+2}\| \ldots \|r_t$. D' gives z to D. Output whatever D outputs.

Note that if r were pseudorandom, i.e. $r = G(s)$ for a random seed s, then $z \sim Z_i$ is distributed according to Z_i; on the other hand, if r were truly random, i.e., $r \leftarrow \{0,1\}^{n+1}$, then $z \sim Z_{i+1}$ is distributed according to Z_{i+1}.

Therefore

$$\begin{aligned}
&\left| \Pr_{s\leftarrow\{0,1\}^n, r=G(s)}[D'(r)=1] - \Pr_{r\leftarrow\{0,1\}^{n+1}}[D'(z)=1] \right| \\
=&\left| \Pr_{z\leftarrow Z_i}[D(z)=1] - \Pr_{z\leftarrow Z_{i+1}}[D(z)=1] \right| \\
\geq& 1/\operatorname{poly}(n)\,.
\end{aligned}$$

□

Reflection on hybrid arguments

- Extreme hybrid distributions match the original cases of interest.
- Distinguishing consecutive hybrids implies breaking some underlying assumption.
- The number of hybrids should be polynomial.

Key distribution and management

We've been dodging a critical question for a long time:

How do the users share a *secret* key in the first place?

We can think of arranging secure *in-person meetings* to distribute secret keys from time to time. Alternatively, users may be able to use some trusted courier service. In both settings, we are essentially assuming existence of some *secure channels* available.

But, if a secure channel is present, why bother with private-key crypto at all? Well the point here is that these forms of secure channels are either only available for a limited period of time, or are so costly and inefficient. Hence, while government and military agencies might afford such a solution, it is completely not feasible at a large-scale, e.g. on the Internet. For example, if there are N users on the Internet, to enable pair-wise secure communication, we would need to arrange $\Omega(N^2)$ meetings to distribute secret keys. Meanwhile, storing and managing a large number of keys by the users is difficult and prone to attacks.

The concerns above in principle can be addressed in a "closed"-system, where there is a well-defined population of users and they are willing to (and are capable of) following the same policies for distributing and storing keys. However, there is another issue in "open" systems which is actually more common once told. Consider using encryption to send credit-card information to an Internet merchant to complete a transaction for the very first time. In such transient interactions, one may not be aware of the other's existence until the time they want to communicate securely.

To summarize, there are at least three issues regarding the use of private-key cryptography:

i) how to distribute secret keys?

ii) how to store and manage large number of keys?

iii) how to handle dynamic interactions in open systems?

A partial solution: key distribution centers. Basic idea: a centralized entity manages adding/removing new users, maintaining and issuing session keys.

Read [KL: 10.2] more details. Draw diagram

Advantages of KDC

- Each user needs to store only one long-term key (shared with the KDC). Sessions keys are short-term and are erased once a communication session concludes.
- Adding a new user amounts to setting up a key between new user and KDC.

Limitations of KDC

- Much workload on KDC.
- KDC is a single point of failure: if KDC is down (system error or intentional attacks), the entire system is unavailable or completely compromised.

Public-key revolution

We still do not have a solution to distribute secret keys from scratch. In fact, it seems inevitable that some form of secure channel must be established beforehand. This is a common belief in the long history of cryptography (i.e. secret writing), which apparently needs no further justification. It is until about half a century ago that people started to challenge this belief and think out of the box.

The few pioneers in the public domain are Ralph Merkle, Whitfield Diffie, Martin Hellman, Ron Rivest, Adi Shamir, Leonard Adelman, and following them Goldwasser and Micali. Crytpo is never just a civil business. There was also independent (earlier) development in the British intelligence agency GCHQ. A short story comes later.

The inspiration came from an ingenuous idea for communicating secretly over the phone line during the WWII. The receiver would first inject some noise on the phone line, and then the sender transmits its signal. Since it is mixed with the receiver's noise, no eavesdropper can learn the sender's signal, but the receiver can clean up the noise and recover the intended signal. Note that the sender and receiver need not meet in advance to share secret information at all.

What's the distinguishing feature in this process compared to private-key encryption? The receiver initiates the process! Intuitively, here is the essence of this idea: the receiver prepares a box with a lock on it, which he and only he possesses a key that can open the lock;. Then the box is mass produced and placed at a public domain. Anyone can grab a box, and lock some message inside. Clearly, only the receiver can open it with the key (unless you violently smash the lock...)

The early visionaries of PKC then envisioned an abstract "magic" object to capture the asymmetry between the receiver and the sender (we call *trapdoor one-way permutations* these days): a function (or rather a family of functions) F which takes two inputs. Any one can generate a pair of keys (pk, sk), a public key and a secret key or trapdoor of the public key; and they define two functions $F(pk, \cdot)$ and $F(sk, \cdot)$, and $F(sk, \cdot) = F^{-1}(pk, \cdot)$. The properties we'd like are

- $F(pk, \cdot)$ and $F(sk, \cdot)$ can be computed efficiently;
- $F(pk, \cdot)$ is hard to invert without knowing sk.

With this, we can encrypt in a *public* way: suppose Alice can publish pk so that anyone who wants to communicate with her can just encrypt by computing $c := F(pk, m)$, which Alice can decode by inverting using sk, i.e., $m := F(sk, c)$. In particular, Alice and Bob can transmit a key for a private-key scheme now via purely public communication! Sharing a key is resolved!

They also suggested that the reverse process can actually serve as an *authentication* mechanism where Alice authenticates with sk by computing $t = F^{-1}(sk, m)$, and anyone knowing pk can verify the message integrity (as well as identity authentication). This is called digital signature.

This announces the dawn of public-key encryption and digital signature, in the seminal paper[3].

In fact, suggested by Ralph Merkle, Diffie and Hellman also proposed another approach for establishing a shared secret key with a public channel, this is the famous Diffie-Hellman key-exchange protocol that we will study later.

[3] Diffie, Whitfield, and Martin Hellman. "New directions in cryptography." IEEE transactions on Information Theory 22.6 (1976): 644-654. `https://www-ee.stanford.edu/~hellman/publications/24.pdf`

Defining public-key encryption

Definition 4. A public-key encryption scheme is a triple of PPT algorithms (G, E, D) such that

1. G: $(pk, sk) \leftarrow G(1^n)$. pk: public-key, sk: secret-key.
2. E: on input pk and m, $c \leftarrow E_{pk}(m)$.
3. D: takes sk and c, and computes $m := D_{sk}(c)$ (assuming D is always deterministic).

Correctness requirement: $D_{sk}(E_{pk}(m)) = m$ for any m except with negligible probability.

Did you notice the main distinction from a private-key encryption scheme? G generates a pair of keys (pk, sk): one for encryption and the other one for decryption. [4]

[4] That's why people aslo call them asymmetric encrytion and symmetric encrytion respectively

The standard security notion for PubKE will be CPA security, and the key idea should be familiar to you by now. Given a PubKE scheme $\Pi = (G, E, D)$ and an adversary $\mathcal{A}$, consider

1. CH runs $(pk, sk) \leftarrow G(1^n)$.
2. $\mathcal{A}$ is given pk, and output (m_0, m_1) with $|m_0| = |m_1|$.
3. CH picks uniform bit $b \leftarrow \{0, 1\}$ and computes $c \leftarrow E_{pk}(m_b)$. Send this challenge ciphertext c to $\mathcal{A}$.
4. $\mathcal{A}$ outputs b'. $\mathcal{A}$ succeeds if $b' = b$, and let $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$ in this case. Otherwise define $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 0$.

Figure 1: The CPA indistinguishability game $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

Definition 5. A public-key encryption scheme Π is CPA-secure if for any PPT $\mathcal{A}$,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)\,.$$

Where is the CPA part? Once pk is given, $\mathcal{A}$ can just implement the encryption oracle E_{pk} on its own. Hence in the public-key setting, there makes no difference between an *eavesdropper* and an *CPA-adversary*, in contrast to the private-key setting (computational secrecy is strictly weaker than CPA-security).

Again, CPA-security necessarily needs randomized encryption.

Theorem 6 ([KL: Thm.11.4])**.** *No determinstic public-key encryption is CPA-secure.*

We do not formally discuss CPA-security for encrypting multiple messages [KL: Definition 11.5] other than stating the important fact

1-bit encryption is complete for public-key CPA-security.

We will spend the next few lectures constructing PKC. We need some tools from number theory. Read the relevant sections on the textbook and also the note by Trevisan posted on our course webpage.

Review HW2/Quiz2

Fun reading: early days of PKC

Here is my short, incomplete and perhaps biased version of the story happened at the dawn of public-key cryptography. [5]

In the public domain, Ralph Merkle was probably the first to propose a brand new approach to key distribution. In Fall 1974, Merkle was then an undergrad at UC Berkeley, and he wrote in his project proposal [6] of a computer security course that

> "it might seem intuitively obvious that if two people have never had the opportunity to prearrange an encryption method, then they will be unable to communicate securely over an insecure channel... I believe it is false."

However, his proposal drew little interest from the professor who responded "not good enough". Merkle later dropped the class and continued working on it. Encouraged by another faculty member at Berkeley, he submitted a draft to the Communications of the ACM in August of 1975, but only to get comments as follows:

> "I am sorry to have to inform you that the paper is not in the main stream of present cryptography thinking and I would not recommend that it be published in the Communications of the ACM."
>
> "Experience shows that it is extremely dangerous to transmit key information in the clear."

No doubt, his paper got rejected. This only confirmed Merkle that no one had previously investigated this approach. After a long delay, his paper finally got published [7] in 1978.

In essence what Merkle proposed was a protocol for two parties to exchange a secret key over a public (insecure) communication channel by using a publicly accessible hash function $\mathcal{O}$ (i.e. in modern term, it works in the random-oracle model). He showed that

- honest parties will agree on a key with N queries to $\mathcal{O}$.
- any eavesdropper has to spend $\Omega(N^2)$ queries to $\mathcal{O}$ to leran the key.

He conjectured that "it might be possible to obtain a protocol where breaking is exponentially harder than using them". But no concrete candidates were given.

History often shows the same pattern indicating the beginning of a *paradigm shift*. There were other people at the same time (and more to come) who envisioned innovative approaches to cryptography beside Merkle. The most influential were the two visionaries at Stanford then, Diffie and Hellman.

[5] This account is main based on the following sources:

- The first ten years of PKC by WitField Diffie `http://cr.yp.to/bib/1988/diffie.pdf`.
- Ralph Merkle's self account of his CS244 course project in 1974 at UC Berkeley `http://www.merkle.com/1974/`.
- THE STORY OF NON-SECRET ENCRYPTION by J. H. Ellis at CESG `http://cryptome.org/jya/ellisdoc.htm`.

[6] The original CS244 project proposal from Fall of 1974 (7 page PDF) `http://www.merkle.com/1974/FirstCS244projectProposal.pdf`. A two-page version `http://www.merkle.com/1974/SecondCS244projectProposal.pdf`.

[7]

They realized the need for a revolution in cryptography that goes beyond the conventional domain of intelligence and military applications to a public environment such as commercial applications due to the "development of cheap digital hardware". They envisioned a new type of cryptography inspired by the "asymmetric" phenomenon in the physical world. Basically they imagined a magic box (we call them *trapdoor one-way permutations* these days), which is a collection of permutations $\{F_k\}$ which is easy to compute $F_k(x)$ but hard to invert $F_k^{-1}(y)$. But for each k, if some secret information $sk(k)$ (a *trapdoor*) is known then inverting becomes easy $x = F^{-1}(sk(k), y)$. With this, any can encrypt in a *public* way: suppose Alice knows $sk(k)$, then she can publish k so that anyone who wants to communicate with her can just encrypt by computing $F_k(m)$, which Alice can decode by inverting using $sk(k)$. They also suggested that the reverse process can actually serve as an *authentication* mechanism where Alice authenticates with $sk(k)$ by computing $t = F^{-1}(sk(k), m)$, and anyone knowing k can verify the integrity (as well as identity authentication).

This marked the invention of public-key *encryption* and *digital signature*. However, they didn't know how to instantiate such a magic box F. Later they met Merkle, and got inspired by Merkle's idea of key exchange. They were lucky to receive a suggestion of a mathematical tool from a Stanford colleague, and they came up with the famous *Diffie-Hellman* key-exchange protocol. This concrete protocol and their ingenious conceptual introduction of public-key cryptography were finally compiled in this ground-breaking paper "New Directions in Cryptography" [8]. In sum, their main contributions are [8]

- Introducing **public-key cryptography**, including the notion of public-key *encryption* and *digital signature*, and an approach based on a magic box *trapdoor permutations*.
- Proposing the **DH key-exchange protocol** based on a number-theoretical problem which achieves the exponential gap that Merkle conjectured, if one is willing to accept certain assumptions on the computational hardness of the number-theoretical problem.

Finding a proper candidate for the magic box had to wait for another year by another three pioneers Rivest, Shamir and Adelmen [9], [9] who introduced the famous RSA functions. We hence have concrete public-key cryptosystems since.

Discovery at GCHQ. Interesting enough, according to a declassified document at the British intelligence agency GCHQ in 1997, similar ideas were developed even before the discovery in the public research community. Apparently in late 1960's, James Ellis envisioned basically the same "magic" box that Diffie-Hellman later proposed and thought

about using it for a new type of encryption – public-key encryption. He didn't know how to implement such a box either. He kept assigning this problem to new recruits until in 1973/74, Clifford Cocks and Malcolm Williamson came up with a solution essentially the same as RSA.