CSE 526A: Cryptography

University of Washington

Andrew Chen

Spring 2026

Hello and welcome! This is my lecture notes on CSE 526A: Cryptography. This is a graduate-level cryptography course intended for MS students, which covers many "modern-day" cryptographic theory and concepts. The professor is **Nirvan Tyagi**, and we meet MW at **10:00 am** for lectures. There are no required textbooks for this course, although we sometimes refer to **Rafael Pass and Abhi Shelat's A Course in Cryptography**. Also note that theorem names might not necessarily be accurate; it's probably just whatever my textbook / professor said it is.

The goal of these lecture notes is to write **understandable** math. As the great Albert Einstein put it, "If you can't explain it to a six year old, then you don't understand it yourself". The hope is that anyone coming across these notes (like you!) will be able to at least take away the gist of these concepts. Should you find any errors in my mathematics, please contact me at zchen66@uw.edu

Contents

List of Definitions

List of Theorems

1 Lecture 01: Mar. 30th

Course overview. Cryptographic object defined both in terms of syntax and security. Security defined via a (1) Game and (2) Advantage metric measuring success of the advesary in the game. Security = all feasible adversaries only win game with negligible probability.

1.1 Course logistics

As seen in the notes introduction, this course is a graduate level "introduction" to cryptography, with a focus on rigorous math and theory. The goal of this course is to prepare students for actual cryptographic research upon completion. Students will be assessed in 2 ways – bi-weekly problem sets (of which there are 4), and a take-home final exam.

The topic breakdown of the course is as follows:

- **Basic cryptographic "primitives"**: hash functions, pseudorandom functions, message authentication codes
- **Encryption**: secret-key encryption, public-key encryption, underlying number theoretic problems
- **Digital signatures**
- **Advanced cryptography**: lattice-based & post-quantum cryptography, multiparty computation, zero-knowledge proofs

1.2 Provable security

To begin, we discussed a bit about why build secure systems is hard, and why we need cryptography in the first place. Just think about it, strictly speaking, the adversary's strategy set is *infinite*, and we don't really have a way to "test" a sytem against all possible adversarial strategies.

How designing for security usually goes is: some prototype of a system is proposed, someone discovers a potential attack, the system is redesigned, another potential attack is discovered, etc etc. Until some version of the system is believed to be "pretty secure", where it's officially released.

So this is bad. Because an attack can still be found for the officially released system. This is why we need **provable security**.

Modern cryptography establishes this concept using math. In order to do define provable security, we'd need some precise definitons as to what the attaker does and what's considered a "successful" attack. This is done through the lens of complexity theory.

To dive a bit deeper into what provable security theorems look like,

"Theorem." Comuputational problem A is intractable *implies* it is intractable to break [Security property] of Scheme B.

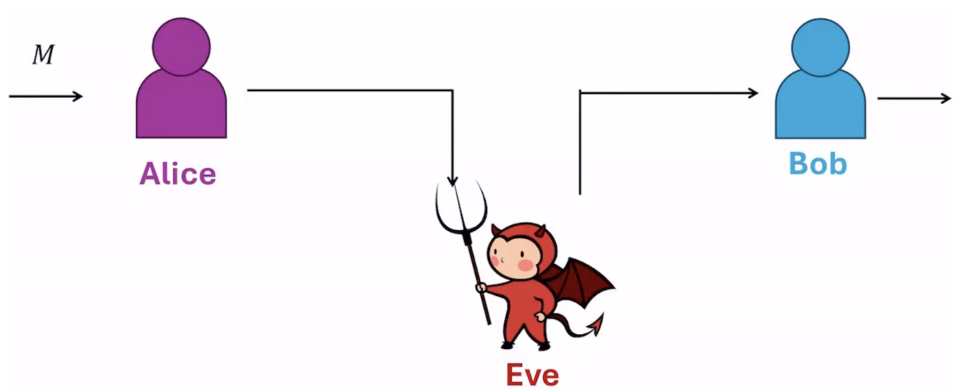
Some examples of what "computational model A" could look like are: integer factorization, breaking AES block ciphers, etc. So all of this means nothing if A isn't intractable.

And typically, these theorems are proven through "contraposition", by supposing that there exists an algorithm that breaks the security property of B, then showing that that algorithm relies on the tractability of A. This strategy is called a **reduction**.

We can take a look at this with a (not-so) simple motivating example.

1.3 Message Integrity

Here's the sitch: alice wants to send a message reliably to bob through some channel *fully controlled* by eve. In other words, when bob receives a message, this message must have come from alice and have not been altered by eve.



Looking at this, it's pretty trivial for eve to intercept some message M and interchange it for some other message M'. To make this secure, we can introduce a bit of cryptography.

Suppose now alice and bob both have access to some secret key K that eve does not have. Then, alice can run some public keyed fuction $F(K, M) = T$ and send both the message M and the tag T to bob.

From bob's perspective, he will have received a mesage-tag pair M', T', at which point he will only accept the received message M' if $F(K, M') = T'$.

Intuitively speaking, the security property we're looking for is that eve shouldn't be able to generate a pair M', T' such that $F(K, M') = T'$. Formally, this is called a **Message Authentication Code (MAC)**. Let's formalize this cryptographic object – both the *syntax* and *security*.

Definition (Message authentication code (MAC))**.** A **MAC** is a keyed function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. The first argument is referred to as the "key", and the second argument is referred to as the "message". We are require F to be efficiently computable (which we will discuss later).

Now for the security property we're targeting is called **Unforgeability under Chosen Message Atttack (UF-CMA)**:

- Adversary is not given key K
- Even if adversary learns $F(K, M_i)$ for chosen messages $M_1, M_2, ...$, the adversary cannot predict $F(K, M)$ for some $M \notin \{M_1, M_2, ...\}$.

Note the wording of "chosen" messages above – this means we assume the tags eve sees are from messages that she is choosing, not only just the ones sent by alice. This is a reasonable assumption to have because (1) it gives more power to the advesary which requires stronger security from our end, and (2) it's entirely realistic irl because adversaries can influence the message sender.

But let's back up for a second and formalize the concept of an "adversary". In cryptography, adversaries are almost always algorithms, and in order to measure how much these algorithms succeed in breaking cryptographic schemes, we define a metric called **advantage**.

For UF-CMA, for example, we denote this by

$$\mathsf{Adv}_F^{\texttt{uf-cma}}(\mathcal{A}) \in [0, 1]$$

this tells us "how well" adversary $\mathcal{A}$ breaks the `UF-CMA` security of F, on a scale of 0 - 1. To circle back to our original problem, our goal is to say that $\mathsf{Adv}_F^{\texttt{uf-cma}}(\mathcal{A})$ is *small* for all *feasible* $\mathcal{A}$.

But what is "small"? What is "feasible"? These are formalities we need to eventually define.

In literature language, we define $\mathsf{Adv}_F^{\texttt{uf-cma}}(\mathcal{A}) \in [0, 1]$ as the probability of winning a **game**. We will be formally defining these "games" using pseudocode.

Through these games, we specify what $\mathcal{A}$ can do: what inputs $\mathcal{A}$ is given, and what operations $\mathcal{A}$ can perform in the form of **oracles**. Let's take an un-abstracted look at this now:

Algorithm 1 UF-CMA Game

Game UF-CMA$_F(\mathcal{A})$
  $K \xleftarrow{\$} \mathcal{K}$
  $\mathcal{Q} \leftarrow \emptyset$
  $(X^*, T^*) \leftarrow \mathcal{A}^E()$
  return $(X^* \notin \mathcal{Q} \wedge F(K, X^*) = T^*)$

Oracle E(X)
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X\}$
  return $F(K, X)$

Now that we have this algorithm set in stone, we can define the advantage to be

$$\mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{A}) = \Pr[\text{UF-CMA}_F(\mathcal{A}) \Rightarrow \text{true}]$$

Let's do a thought experiment. Suppose $F(K, X) = K \oplus X$. Is F UF-CMA secure?

Well consider the following adversary:

Advesary $\mathcal{A}^{\mathbf{E}}()$:
$\bar{K} \leftarrow \mathbf{E}(0^n)$
$X^* \leftarrow 1^n$
$T^* \leftarrow \bar{K} \oplus 1^n$
return (X^*, T^*)

This adversary will *always* guess the correct answer, and thus will have advantage 1.

Remark. We will never feasibly achieve an advantage of 0.

Why is this? Because if the adversary were to just guess random tags sampled from the tag space $\mathcal{Y}$, then eventually it will guess a correct tag for some chosen message X^*.

2 Lecture 02: Apr. 1st

Limits on adversary advantage. Notions of security. Concrete vs asymptotic security. Multiple security properties. UF-CMA and UF-CMA* equivalence.

2.1 Efficiency & Negligibility

Recall the UF-CMA security criterion we've decided on last time – eve should not be able to forge the tag for some message M that she has not yet seen.

But there was a slight problem with the way we've formalized this statement. We said that "F is UF-CMA secure if $\mathsf{Adv}_F^{\texttt{uf-cma}}(\mathcal{A})$ is *small* for all *feasible* $\mathcal{A}$'s." What is "small"? What is "feasible"?

Consider the following adversary:

Adversary $\mathcal{B}^{\mathbf{E}}()$
    $T^* \xleftarrow{\$} \mathcal{Y}$
    return (X^*, T^*)

This is a guessing adversary. We talked a bit about this last time, but this adversary could just "get lucky" and win the game, with probability $\frac{1}{|\mathcal{Y}|}$.

Going back to the previous point. As a consequence of this generic attack, in order to make the advantage "small", we would need to require $|\mathcal{Y}| \gg 0$ (typically $|\mathcal{Y}| = 2^{128}$).

Advesary $\mathcal{B}$ is fast, but only succeeds with small-ish probability. Let's take a look at another (rough) adversary idea – one that has high advantage, but may also take a long time to run:

1. Make a few queries, learns $F(K, X_i)$ for $X_1, \ldots, X_q$
2. Find K' such that $F(K', X_i) = T_i$ for all $i \in [q]$
3. Return $(X^*, F(K', X^*))$ for $X^* \notin X_1, \ldots, X_q$

Will this always work? Not necessarily. Notice $F(K', X_i) = F(K, X_i) = T_i$ for all $i \in [q]$ does not imply $F(K', X^*) = F(K, X^*)$. This is due to the existance of "spurious" keys, where for each $X_i \in \mathcal{X}$, there could exist many K that map it to T_i.

Contrary to the first adversary $\mathcal{B}$, this adversary (we'll call $\mathcal{G}$) achieves very large advantage, but with time proportional to $|\mathcal{K}|$. So again, this would require us to have a sufficiently large key space $|\mathcal{K}| \gg 0$.

With both these attacks $\mathcal{B}$ and $\mathcal{G}$, we have established that we need to make our key and tag spaces "large". But how large is "large" exactly?

*****

To answer that previous question, we'll need some formalized notions of feasibility and negligibility. We start with a definition.

Definition (randomized algorithm running time)**.** A randomized algorithm A has running time $t : \mathbb{N} \to \mathbb{N}$ if on input x, the algorithm $A(x)$ runs for at most $t(|x|)$ steps, independently of its random choices.

> **Intuition.** *t is a function that maps input size to running time.*

If A has polynomial running time $t(n) = O(n^c)$, we refer to A as "PPT" (=Probabilistic Polynomial Time).

Given this definition now, one thing we could say about the feasibility of our MACs is to limit them to be PPT. But polynomial time in what exactly?

One solution is to **parameterize** our algorithms. To be more psecific, we think of a MAC as parameterized by some factor $\lambda \in \mathbb{N}$:

$$F : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{l(\lambda)}$$

Formally, we should think of $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ as a family of functions parameterized by λ, where each function $F_\lambda : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{l(\lambda)}$.

Let's update our UF-CMA definition:

Algorithm 2 Parameterized UF-CMA Game

```
Game UF-CMA_F(A, λ)
    K ←$ {0,1}^λ
    Q ← ∅
    (X*, T*) ←$ A^E(1^λ)
    return (X* ∉ Q ∧ F_λ(K, X*) = T*)

Oracle E(X)
    Q ← Q ∪ {X}
    return F_λ(K, X)
```

One thing to note is the input to the adversary $\mathcal{A}$, which signifies that $\mathcal{A}$ is PPT in $|1^\lambda| = \lambda$.

> **Intuition.** *All of this is just a fancy way to define MACs (and thus adversaries) to be PPT with respect to the key-length.*

With this idea of λ emerges two philosophies - **concrete security** and **asymptotic security**.

In concrete security, security is not treated as $\{0,1\}$, some schemes are more secure than others, and that is determined by a lot of counting. This matters in picking the right λ and is more practice-friendly.

On the other side, in asymptotic security, a system is treated as either secure or not secure. It's easy to state theorems, and picking the right λ does not matter. This is much more theory-friendly.

Now back to defining security. Let's define UF-CMA, concrete edition! Let $t : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to [0, 1]$ to be functions.

Definition (Concrete UF-CMA)**.** A family $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ is (t, ϵ)**-UF-CMA secure** if

$$\mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{A}, \lambda) \leq \epsilon(\lambda)$$

for all randomized attackers $\mathcal{A}$ with running time t.

For example, $(2^{0.5\lambda}, 2^{-0.3\lambda})$-UF-CMA secure is better than $(\lambda^2, \frac{1}{\lambda^5})$-UF-CMA secure. The first case gives adversaries the ability to run in exponential time with respect to the input size and less leniency in advantage bounds, whereas the second case puts more strict limits on the adversary, and thus it is more secure.

Similarly, we have for the asymptotic case:

Definition (Asymptotic UF-CMA)**.** A family $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ is **UF-CMA secure** if for all polynomials t there exists a **negligible** function ϵ such that F is (t, ϵ)-UF-CMA secure.

Definition (Negligible functions)**.** A function $\epsilon : \mathbb{N} \to [0, 1]$ is **negligible** if for all $c \in \mathbb{N}$ there exists $\lambda_0(c) \in \mathbb{N}$ such that $\epsilon(\lambda) \leq \frac{1}{\lambda^c}$ for all $\lambda \geq \lambda_0(c)$.

To end off our efficiency conversation, in this class, theorem proofs and ideas are largely the same across both the concrete and asymptotic cases.

2.2 Reductions

We introduce another game:

Algorithm 3 KR-CMA Game

Game KR-CMA$_F(\mathcal{A}, \lambda)$
    $K \xleftarrow{\$} \{0, 1\}^\lambda$
    $K^* \leftarrow \mathcal{A}^E(1^\lambda)$
    return $(K^* = K)$

Oracle E(X)
    return $F(K, X)$

The advantage is defined

$$\mathsf{Adv}_F^{\text{kr-cma}}(\mathcal{A}, \lambda) = \Pr[\text{KR-CMA}_F(\mathcal{A}, \lambda) \Rightarrow \text{true}]$$

This is another security criterion called Key Recovery under Chosen Message Attack. How do we show that this is secure?

We bring back our good friend UF-CMA. Hopefully the similarities are striking.

Theorem 2.1.

For every kr-cma adversary $\mathcal{A}$, there exists an uf-cma adversary $\mathcal{B}$ such that

$$\mathsf{Adv}_F^{\textit{kr-cma}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_F^{\textit{uf-cma}}(\mathcal{B}, \lambda)$$

Moreover, $\mathcal{B}$ is roughly as efficient as $\mathcal{A}$.

Corollary 2.1.1.

If F is (t, ϵ)-UF-CMA secure, then F is (t, ϵ)-KR-CMA secure.

Proof. Our first proof! This one is relatively straight-forward.

By definition of UF-CMA security, we know $\mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}) < \epsilon$ for all $\mathcal{B}$ with runtime $< t$. By the inequality established in the theorem above, we then know

$$\mathsf{Adv}_F^{\text{kr-cma}}(\mathcal{A}) \leq \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}) < \epsilon$$

For all $\mathcal{A}, \mathcal{B}$ with runtime $< t$. This by definition means F is (t, ϵ)-KR-CMA secure. $\square$

Very similarly, in the asymptotic case,

Corollary 2.1.2.

If F is UF-CMA secure, then F is KR-CMA secure.

Now that we've established these corollaries on the security notion of KR-CMA, we should actually show the implication UF-CMA $\Rightarrow$ KR-CMA. This is our first *big* proof.

Proof. Consnider the following UF-CMA adversary $\mathcal{B}$:

Algorithm 4 UF-CMA $\Rightarrow$ KR-CMA Reduction

Adversary $\mathcal{B}^{\mathbf{E}}(1^\lambda)$
  $\mathcal{S} \leftarrow \emptyset$
  $K' \xleftarrow{\$} \mathcal{A}^{\mathbf{E'}}(1^\lambda)$
  Pick $X^* \in \mathcal{X} \setminus \mathcal{S}$
  $T^* \leftarrow F(K', X')$
  return (X^*, T^*)

Oracle E'(X)
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{X\}$
  return $\mathbf{E}(X)$

Here $\mathcal{A}$ is a KR-CMA adversary. Notice that $\mathcal{B}$ perfectly simulates the environment of a KR-CMA game for $\mathcal{A}$. This means if $\mathcal{A}$ outputs the correct key, then (X^*, T^*) is a valid message-tag pair, and $\mathcal{B}$ wins the UF-CMA game.

Now suppose $\mathcal{A}$ achieves high advantage in the KR-CMA game. This would directly imply that $\mathcal{B}$ would achieve just as high advantage in the UF-CMA game. In other words,

$$\mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}, \lambda) \geq \mathsf{Adv}_F^{\text{kr-cma}}(\mathcal{A}, \lambda)$$

This proves the desired statement. $\square$

Now let's take a thought experiment. What if we wanted to prove KR-CMA $\Rightarrow$ UF-CMA? Intuitively, this is untrue. But how do we show it?

We can prove this via a counter-example – a function that's KR-CMA secure but not UF-CMA. Consider $F(K, X) = X$. Here, F is definitely KR-CMA, because there's no way for us to recover K without bruteforcing it.

However, F is very trivial to be forged, as we can just choose any X^*, and we then know that the correpsonding tag is just itself.

Thus far, we've pretty thoroughly discussed this UF-CMA security criteria for MACs. But there are actually many other security criterias out there, it just happened so that UF-CMA is the popular one that emerged through literature.

One other security criteria we can have is a spin-off of the traditional UF-CMA. Consider the following game:

Algorithm 5 UF-CMA* Game

Game UF-CMA$^*_F(\mathcal{A}, \lambda)$
    $K \xleftarrow{\$} \{0,1\}^\lambda$
    $\mathcal{Q} \leftarrow \emptyset$
    win $\leftarrow$ false
    Run $\mathcal{A}^{\mathbf{E},\, \mathbf{V}}(1^\lambda)$
    return win

Oracle $\mathbf{E}(X)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X\}$
    return $F(K, X)$

Oracle $\mathbf{V}(X, T)$
    if $X \notin \mathcal{Q} \wedge F(K, X) = T$ **then**
        win $\leftarrow$ true
        return true
    return false

The advantage, not too surprisingly, is defined

$$\mathsf{Adv}_F^{\text{uf-cma}^*}(\mathcal{A}, \lambda) = \Pr[\text{UF-CMA}^*_F(\mathcal{A}, \lambda) \Rightarrow \text{true}]$$

Intuitively, this game is simply the traditional UF-CMA game, except now the adversary $\mathcal{A}$ gets multiple guesses. The new oracle $\mathbf{V}$ is a verification oracle that checks if $\mathcal{A}$ has successfully succeed in the game.

Now the question begs to be asked: how does this relate to UF-CMA? We have the following theorem:

Theorem 2.2.
For every uf-cma adversary $\mathcal{A}$ making at most q_V queries to $\mathbf{V}$, there exists a uf-cma adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_F^{\textit{uf-cma}^*}(\mathcal{A}, \lambda) = q_V \cdot \mathsf{Adv}_F^{\textit{uf-cma}}(\mathcal{B}, \lambda)$$

Moreover, $\mathcal{B}$ is roughly as efficient as $\mathcal{A}$.

Corollary 2.2.1.
If F is (t, ϵ)-UF-CMA secure, then F is $(t, q_V \cdot \epsilon)$-UF-CMA secure.*

> **Intuition.** *Basically, UF-CMA* is a stronger security definition than UF-CMA, since an adversary for the UF-CMA* game will have multiple guesses. This corollary tells us what happens when we take a scheme proven secure under the standard UF-CMA definition and subject it to this stronger UF-CMA* adversary: their probability of success grows, but we can mathematically bound their advantage to at most $q_V \cdot \epsilon$.*

Let's now prove this statement where (t, ϵ)-UF-CMA $\Rightarrow$ $(t, q_V \cdot \epsilon)$-UF-CMA*.

Proof. Consider the following reduction:

Algorithm 6 UF-CMA $\Rightarrow$ UF-CMA* reduction

Game $\mathrm{G}(\mathcal{A}, \lambda)$
    $K \xleftarrow{\$} \{0,1\}^\lambda$
    $\mathcal{Q} \leftarrow \emptyset$
    $\text{win} \leftarrow 0$
    $c \leftarrow 0$
    Run $\mathcal{A}^{\mathbf{E},\,\mathbf{V}}(1^\lambda)$
    return win   # *index of first win*

Oracle $\mathbf{E}(X)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X\}$
    return $F(K, X)$

Oracle $\mathbf{V}(X, T)$
    if $X \notin \mathcal{Q}$ **then**
        $c \leftarrow c + 1$
        if $F(K, X) = T$ **then**
            if $\text{win} = 0$ **then** $\text{win} \leftarrow c$
            return true
    return false

Essentially, rather than returning a $0/1$ value for whether or not $\mathcal{A}$ has succeeded, we instead return the *index* of the first successful guess by $\mathcal{A}$.

Before we go any further, let's establish a lemma.

Lemma 2.3.

For every $i \in [q_V]$, there exists a uf-cma adversary $\mathcal{B}_i$ such that

$$\Pr[G(\mathcal{A}, \lambda) \Rightarrow i] = \mathsf{Adv}_F^{\textit{uf-cma}}(\mathcal{B}_i, \lambda)$$

Moreover, $\mathcal{B}_i$ is roughly as efficient as $\mathcal{A}$.

> **Intuition.** *Essentially, each guess i of $\mathcal{A}$ in UF-CMA* can be represented by an independent adversary $\mathcal{B}_i$ for the standard 1-guess UF-CMA.*

Coming back to the main proof, we can write the UF-CMA* advantage of $\mathcal{A}$ as

$$\begin{aligned}\mathsf{Adv}_F^{\text{uf-cma}^*}(\mathcal{A},\lambda) &= \sum_{i=1}^{q_V} \Pr[G(\mathcal{A},\lambda) \Rightarrow i] \\ &= \sum_{i=1}^{q_V} \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}_i,\lambda)\end{aligned}$$

Notice how we applied the previous lemma. Before we go further, let's learn another useful technique: *composite adversaries.*

Adversary $\mathcal{B}^{\mathbf{E}}(1^\lambda)$
$\quad i \xleftarrow{\$} [q_V]$
$\quad (X^*, T^*) \leftarrow \mathcal{B}_i^{\mathrm{E}}(1^\lambda)$
$\quad$ **return** (X^*, T^*)

Here, adversary $\mathcal{B}$ is a composite adversary of all $\mathcal{B}_i$, where it simply randomly picks a $\mathcal{B}_i$ and runs it. Furthermore, the advantage of this adversary would be the average of the advantages of $\mathcal{B}_i$.

Using this technique, we can expand the previous equation further and have

$$\begin{aligned}\mathsf{Adv}_F^{\text{uf-cma}^*}(\mathcal{A},\lambda) &= \sum_{i=1}^{q_V} \Pr[G(\mathcal{A},\lambda) \Rightarrow i] \\ &= \sum_{i=1}^{q_V} \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}_i,\lambda) \\ &= q_V \times \frac{1}{q_V} \sum_{i=1}^{q_V} \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}_i,\lambda) \\ &= q_V \times \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B},\lambda)\end{aligned}$$

This proves the desired statement. □

But we were a bit hand-wavy with the lemma we introduced in the middle of that proof, so let's go back and figure out the exact reduction to explicitly write out $\mathcal{B}_i$ as well:

Proof. (sketch)

Consider the following reduction:

Algorithm 7 UF-CMA $\Rightarrow$ UF-CMA* reduction lemma

Adversary $\mathcal{B}_i^{\mathbf{E}}(1^\lambda)$
    $\mathcal{S} \leftarrow \emptyset$
    Run $\mathcal{A}^{\mathbf{E'},\ \mathbf{V'}}(1^\lambda)$
    return [it don't matter]

Oracle E'(X)
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{X\}$
    return $\mathbf{E}(X)$

Oracle V'(X, T)
    if $X \notin \mathcal{S}$ **then**
        $c \leftarrow c + 1$
        if $c = i$ **then** $\mathcal{B}_i$ stops and returns (X, T)
    return false

To make sense of this, recall that each $\mathcal{B}_i$ is an adversary for the standard UF-CMA game. Since this is a reduction, it then must simulate a UF-CMA* game for $\mathcal{A}$. However, this is a difficult task to do as $\mathcal{B}_i$ doesn't inherently have the verification oracle $\mathbf{V}$.

Essentially what this pseudocode is saying is that $\mathcal{B}_i$ will have high advantage whenever $\mathcal{A}$ succeeds in the UF-CMA* game *on the i-th try.*

In other words, and because this is a proof sketch, we can hand-wavy our way through and say that,

$$\Pr[G(\mathcal{A}, \lambda) \Rightarrow i] = \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{B}_i, \lambda)$$

"proving" the desired statement. $\square$

Okay. To end on a summary of today:

- Usually: (1) advantage cannot be 0 (2) advantage cannot be small against computationally *unbounded* attackers
- Two choices of security: concrete vs asymptotic (choice should not impact the essence of what we try to prove)
- Same object (a keyed function) can satisfy multiple properties (ie KR-CMA and UF-CMA*)
- UF-CMA and UF-CMA* equivalence is an example of a rather involved proof (we only proved one direction today, but technically an equivalence)

3 Lecture 03: Apr. 6th

Pseudorandomness, preudorandom functions (PRF), computational security games, indistinguishability games, PRF-UFCMA reduction, range extensions, hybrid arguments.

3.1 Pseudorandom functions

We began lecture with a quick refresher on MACs and the security notions (UF-CMA, UF-CMA*) associated.

Today, we're going to be looking at another notion known as *pseudorandomness*, and the associated hybrid arguments.

UF-CMA is great so far, it guarantees that it is infeasible to compute a new tag T^* for any unseen message X^*. **However**, it doesn't guarantee the adversary in guessing a large part of T^*, as in, not a *full* tag recovery, but a partial one. Pseudorandomness will guarantee that no part of the output can be predicted.

Let's dive straight into the game. Here, $F : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$.

Algorithm 8 PRF Security Game

Game $\mathrm{PRF}_F(\mathcal{A}, \lambda)$
    $K \xleftarrow{\$} \{0,1\}^\lambda$
    $b \xleftarrow{\$} \{0,1\}$
    $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{E}_b}(1\lambda)$
    return $(b' = b)$

Oracle $\mathbf{E}_0(X)$
    return $F(K, X)$

Oracle $\mathbf{E}_1(X)$
    if $T[x] = \perp$ **then**
        $T[x] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
    return $T[X]$

Immediately, we see that this game has a slightly different flavor as compared to previous games we've seen. Rather than predicting some output, adversaries for the game will be trying to "distinguish" whether they're dealing with a random function, or our MAC F.

So how do we define advantage? We can start by trying our traditional advantage definitions, where

$$\mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{A}, \lambda) = \Pr[\mathrm{PRF}_F(\mathcal{A}, \lambda)]$$

But is this a good idea? Consider an adversary that *only* outputs the same bit, *all the time*. Under this definition, such an adversary would have advantage $\frac{1}{2}$, since the only possible outputs of the game are either 0 or 1.

So, that's not great. Clearly not a negligible advantage. So the naive solution is to just subtract $\frac{1}{2}$. Now,

$$\mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{A}, \lambda) = \Pr[\mathrm{PRF}_F(\mathcal{A}, \lambda)] - \frac{1}{2}$$

Great, now the adversary from before has advantage 0. But there's now a new problem: our advantage is centered around 0 with a range of $[-\frac{1}{2}, \frac{1}{2}]$.

Maybe we try a bit harder. Let's try this:

$$\mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{A}, \lambda) = 2 \cdot \Pr[\mathrm{PRF}_F(\mathcal{A}, \lambda)] - 1$$

Okay, so that didn't solve the problem, and now our advantage has range $[-1, 1]$. But let's think for a second here – what does "negative advantage" mean?

Having advantage -1 would mean that $\mathcal{A}$ is *always wrong*, which by extension actually means it's always right, by just guessing the bit-flip! Using this information, we can derive the final expression for advantage:

$$\mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{A}, \lambda) = |2 \cdot \Pr[\mathrm{PRF}_F(\mathcal{A}, \lambda)] - 1|.$$

This whole random-bit business might get confusing in a single game. Alternatively, we can split it into two games and have

Algorithm 9 PRF Security Game

Game $\mathrm{PRF}_{F,b}(\mathcal{A}, \lambda)$
  $K \xleftarrow{\$} \{0,1\}^\lambda$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{E}_b}(1^\lambda)$
  return $(b' = 1)$

Oracle $\mathbf{E}_0(X)$
  return $F(K, X)$

Oracle $\mathbf{E}_1(X)$
  if $T[x] = \perp$ **then**
    $T[x] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  return $T[X]$

The basic premise hasn't changed much, but this allows us to redefine advantaged to say

$$\mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{A}, \lambda) = |\Pr[\mathrm{PRF}_{F,1}(\mathcal{A}, \lambda)] - \Pr[\mathrm{PRF}_{F,0}(\mathcal{A}, \lambda)]|$$

To definitively know that they're the same, we need to show that any adversary $\mathcal{A}$ achieves the exact same advantage on both definitions. We will show this in problem set 1.

So why do we even have two different definitions? In the broader picture, we've seen two types of security definitions:

- **Computational security games**: goal is to compute something, advantage is probability (like UF-CMA)
- **Indistinguishability security games**: goal is to *distinguish*, advantage compares with random guessing (like PRF)

3.2 PRF vs UF-CMA

This is a great segue into the relationship between pseudorandomness and UF-CMA. Recall from the beginning of the lecture that pseudorandomness means the adversary should not be able to guess any part of our output.

Intuitively, this means PRF should imply UF-CMA. Take a look at the following theorem:

Theorem 3.1.
For every uf-cma adversary $\mathcal{A}$, there exists a prf adversary $\mathcal{B}$ such that

$$\mathsf{Adv}_F^{\textit{uf-cma}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_F^{\textit{prf}}(\mathcal{B}, \lambda) + \frac{1}{2^{\ell(\lambda)}}$$

Moreover, $\mathcal{B}$ is nearly as efficient as $\mathcal{A}$.

Corollary 3.1.1.
If F is a PRF, then it is also uf-cma secure as long as $\ell(\lambda) = \omega(\log \lambda)$.

> **Intuition.** *Remember, the theorem is the contrapositive of the implication! We're trying to show PRF $\Rightarrow$ UF-CMA, which contrapositionally means any UF-CMA adversary $\mathcal{A}$ can be built on top of, into a PRF adversary $\mathcal{B}$.*

Let's prove this.

Proof. Consider the following two games. Here, recall that $\mathcal{A}$ is some uf-cma adversary.

Algorithm 10 PRF $\Rightarrow$ UF-CMA reduction

Game $\mathrm{G}_b(\mathcal{A}, \lambda)$
  $K \xleftarrow{\$} \{0,1\}^\lambda$
  $\mathcal{Q} \leftarrow \emptyset$
  $(X^*, T^*) \xleftarrow{\$} \mathcal{A}^{\mathbf{E}_b}(1^\lambda)$
  if $X^* \notin \mathcal{Q}$ **then**
    $T \leftarrow \mathbf{E}_b(X^*)$
    return $T = T^*$
  return false

Oracle $\mathbf{E}_0(X)$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X\}$
  return $F(K, X)$

Oracle $\mathbf{E}_1(X)$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X\}$
  if $T[x] = \perp$ **then**
    $T[x] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  return $T[X]$

Based off these games, we have two claims:

Claim 1. $\Pr[G_0] = \mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{A})$.

This should be pretty intuitive – when $b = 0$, $\mathcal{A}$ is given the proper UF-CMA wrapper oracle for our mac F, and we are verifying its output (X^*, T^*) using the same oracle, essentially simulating the UF-CMA game.

Claim 2. $\Pr[G_1] \le \frac{1}{2^{\ell(\lambda)}}$

Here, regardless of the output of $\mathcal{A}$, we are "verifying" it using the random oracle $\mathbf{E}_1$, which will return a uniformly random tag T of length $\ell(\lambda)$. Thus, $T^* = T$ will only return `true` with probability $\frac{1}{2^{\ell(\lambda)}}$.

Put together, these claims imply

$$\begin{aligned}
\mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{A}) &= \Pr[G_0] \\
&= \Pr[G_1] + (\Pr[G_0] - \Pr[G_1]) \\
&\le \frac{1}{2^{\ell(\lambda)}} + (\Pr[G_0] - \Pr[G_1])
\end{aligned}$$

Taking a look at the last line of the chain of inequalities from above, intuitively, G_0 and G_1 are as close as F is secure as a PRF. In other words, if F is a PRF, then G_0 and G_1

should be pretty indistinguishable, since the adversary would not know which game is being simulated.

Let's tighten the rigor on this intuition.

Claim. There exists an adversary $\mathcal{B}$, nearly as efficient as $\mathcal{A}$, such that

$$\Pr[G_0] - \Pr[G_1] \le \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B}, \lambda)$$

We now build an adversary to show this.

Algorithm 11 PRF $\Rightarrow$ UF-CMA reduction (cont'd)

Adversary $\mathcal{B}^{\mathbf{E}}(1^\lambda)$
    $\mathcal{Q} \leftarrow \emptyset$
    $(X^*, T^*) \xleftarrow{\$} \mathcal{A}^{\mathbf{E'}}(1^\lambda)$
    if $X^* \notin \mathcal{Q}$ **then**
        $T \leftarrow \mathbf{E'}(X^*)$
        return $T = T^*$
    return false

Oracle $\mathbf{E'}(X)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X\}$
    return $\mathbf{E}(X)$

Here, the given oracle $\mathbf{E}$ would be determined by the PRF game, being either a wrapper for F, or a random function. Now comes a neat trick: notice how $\mathcal{B}$ is essentially a big chunk of the games G_b? We could almost rewrite informally that

Game $\mathrm{G}_b(\mathcal{B}, \lambda)$
    $K \xleftarrow{\$} \{0,1\}^\lambda$
    return $\mathcal{B}^{\mathbf{E}_b}(1^\lambda)$

Since $\mathcal{B}$ is a PRF adversary, we can conclude based on this re-write that

$$\Pr[\mathrm{PRF}_{F,0}(\mathcal{B}, \lambda)] = \Pr[G_0] \qquad \Pr[\mathrm{PRF}_{F,1}(\mathcal{B}, \lambda)] = \Pr[G_1]$$

This gives us the final piece of the puzzle for us to conclude

$$\begin{aligned}
\mathsf{Adv}_F^{\text{uf-cma}}(\mathcal{A}, \lambda) &\le \frac{1}{2^{\ell(\lambda)}} + (\Pr[G_0] - \Pr[G_1]) \\
&\le \frac{1}{2^{\ell(\lambda)}} + |(\Pr[G_0] - \Pr[G_1])| \\
&= \frac{1}{2^{\ell(\lambda)}} + |\Pr[\mathrm{PRF}_{F,0}(\mathcal{B}, \lambda)] - \Pr[\mathrm{PRF}_{F,1}(\mathcal{B}, \lambda)]| \\
&= \frac{1}{2^{\ell(\lambda)}} + \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B}, \lambda)
\end{aligned}$$

as desired. □

(Holy proof bro i can't decide if i love or hate crypto proofs – they feel both very intuitive and elegant, but also at the same time very disorganized? idk)

This now begs the question: is the converse true? Does UF-CMA imply PRF security? Intuitively, the answer is no. But how do we show that?

We show this via a simple counter example. Let $F : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ be UF-CMA secure. Now consider some $G : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)+1}$ such that

$$G(K, X) = (F(K, X) \| 1)$$

Claim 1. G is UF-CMA secure. We show this on problem set 1.

Claim 2. G is not a PRF. Consider the following adversary:

```
1: Adversary A^E(1^λ)
2:     (Y, Z) → E(0^{n(λ)})        // Here Z is the last bit of the output
3:     return Z
```

We have that when we're in the random world, $\Pr[\mathrm{PRF}_{G,1}(\mathcal{A}, \lambda)] = \frac{1}{2}$, because the last bit will match some random bit string half the time.

However, when we're in the ideal world, $\Pr[\mathrm{PRF}_{G,1}(\mathcal{A}, \lambda)] = 1$, since we know G will always output a string ending in 1.

3.3 Range extentions & hybrid arguments

"So this kinda sucks. I wish there was a way to build PRF from UF-CMA..." – nobody.

Well good for you, because there is! As a result of [Naor-Reingold, CRYPTO '98], we know that from any UF-CMA secure F, we can build a PRF which outputs only one bit. But how can we get a larger output? Is there a way to "expand" pseudorandomness?

Say we have a PRF $F : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$, but now we want a PRF with a $2\ell(\lambda)$ bit outputs. Naively, we can define $H : \{0,1\}^{2\lambda} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{2\ell(\lambda)}$, where

$$H((K_1, K_2), X) = (F(K_1, X), F(K_2, X))$$

Is H also a good PRF in this case? It turns out, yes.

Theorem 3.2.
For every prf adversary $\mathcal{A}$, there exists a prf adversary $\mathcal{B}$ such that

$$\mathsf{Adv}_H^{prf}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_F^{prf}(\mathcal{B}_1, \lambda) + \mathsf{Adv}_F^{prf}(\mathcal{B}_2, \lambda)$$

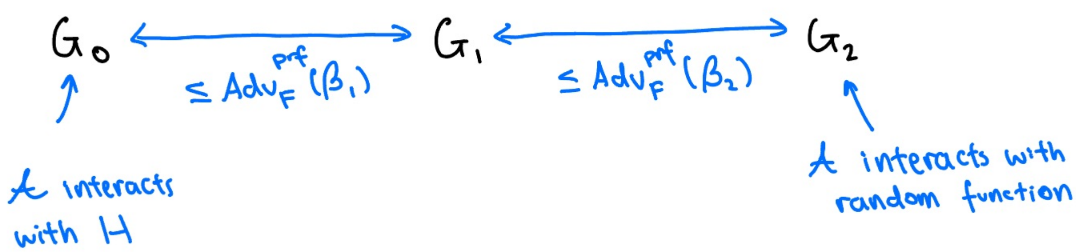
Moreover, $\mathcal{B}_1$ and $\mathcal{B}_2$ are nearly as efficient as $\mathcal{A}$.

Much as expected, we have the associated corollary:

Corollary 3.2.1.
If F is a PRF, then so is H.

Let's prove this!! Yippee!!!! To prove this theorem, we introduce a new technique called **Hybrid argument!**

Proof. Here's the plan. We consider 3 games:



And much like the PRF ⇒ UF-CMA proof from earlier this lecture, we want some adversary $\mathcal{A}$ to not distinguish whether or not they're interacting with G_0 (actual function) or G_2 (random function).

To do this, we want some intermediate game G_1 to use to build our roadmap:

$$\begin{aligned}
\mathsf{Adv}_H^{\mathrm{prf}}(\mathcal{A}) &= \Pr[G_2] - \Pr[G_0] \\
&\leq (\Pr[G_2] - \Pr[G_1]) + (\Pr[G_1] - \Pr[G_0]) \\
&\leq \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B}_1) + \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B}_2)
\end{aligned}$$

Sounds like a plan. Let's get into it. We define the following games, where again, $\mathcal{A}$ is a PRF adversary.

Algorithm 12 Range extension games

Game $G_i()$
  $K_1, K_2 \xleftarrow{\$} \{0,1\}^\lambda$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{O}_i}(1^\lambda)$
  return $(b' = 1)$

Oracle $\mathbf{O}_0(X)$
  return $(F(K_1, X), F(K_2, X))$

Oracle $\mathbf{O}_1(X)$
  if $T_1[x] = \perp$ **then**
    $T_1[x] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  return $(T_1[x], F(K_2, X))$

Oracle $\mathbf{O}_2(X)$
  if $T_1[x] = \perp$ **then**
    $T_1[x] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  if $T_2[x] = \perp$ **then**
    $T_2[x] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  return $(T_1[X], T_2[X])$

Based on these games, we have the following claims.

Claim 0. $\Pr[\mathrm{PRF}_{H,0}(\mathcal{A}, \lambda)] = \Pr[G_0]$

Claim 2. $\Pr[\mathrm{PRF}_{H,1}(\mathcal{A}, \lambda)] = \Pr[G_2]$

These claims should hopefully be intuitive. When $b = 0$, we simulate the ideal world where $\mathcal{A}$ has access to H as defined (combination of two UF-CMA Fs), and when $b = 1$, $\mathcal{A}$ is working with random functions. Now on top of this, we make two more claims.

Claim 01. There exists an adversary $\mathcal{B}_1$, almost as efficient as $\mathcal{A}$, such that

$$|\Pr[G_0] - \Pr[G_1]| = \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B}_1, \lambda)$$

Claim 12. There exists an adversary $\mathcal{B}_2$, almost as efficient as $\mathcal{A}$, such that

$$|\Pr[G_1] - \Pr[G_2]| = \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B}_2, \lambda)$$

We will prove these in a second. But for now, taking these claims for granted, we actually would have completed the proof:

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{prf}}_{H}(\mathcal{A},\lambda) &= |\Pr[\mathrm{PRF}_{H,0}(\mathcal{A},\lambda)] - \Pr[\mathrm{PRF}_{H,1}(\mathcal{A},\lambda)]| \\
&= |\Pr[G_0] - \Pr[G_2]| \\
&= |(\Pr[G_0] - \Pr[G_1]) + (\Pr[G_1] - \Pr[G_2])| \\
&\leq |(\Pr[G_0] - \Pr[G_1])| + |(\Pr[G_1] - \Pr[G_2])| \\
&= \mathsf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}_1,\lambda) + \mathsf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}_2,\lambda)
\end{aligned}$$

Let's now prove **Claim 01**. Consider the following PRF adversary $\mathcal{B}$ against F:

Algorithm 13 Range extension reduction (Claim 01)

Adversary $\mathcal{B}_1^{\mathbf{E}}(1^\lambda)$
    $K_2 \xleftarrow{\$} \{0,1\}^\lambda$
    $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{E'}}(1^\lambda)$
    return b'

Oracle E'(X)
    $Y \leftarrow \mathbf{E}(X)$
    return $(Y, F(K_2, X))$

What we need to argue now is that $\Pr[\mathrm{PRF}_{F,0}(\mathcal{B}_1,\lambda)] = \Pr[G_0]$ and $\Pr[\mathrm{PRF}_{F,1}(\mathcal{B}_1,\lambda)] = \Pr[G_1]$. This should be immediate upon observation:

Essentially, this adversary "simplifies" the previous games G_i by taking apart exactly half of it. It randomizes exactly *one* of the PRFs used in H's range extension, allowing us to reduce *half* of the previous games into the formalized PRF game. Immediately now, we have

$$|\Pr[G_0] - \Pr[G_1]| = \mathsf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}_1,\lambda)$$

We use a very similar idea to prove **Claim 12**.

Algorithm 14 Range extension reduction (Claim 12)

Adversary $\mathcal{B}_2^{\mathbf{E}}(1^\lambda)$
    $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{E'}}(1^\lambda)$
    return b'

Oracle E'(X)
    if $T_1[x] = \perp$ **then** $T_1[X] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
    $Y \leftarrow \mathbf{E}[X]$
    return $(T_1[X], Y)$

□

4 Lecture 04: Apr. 8th

PRF constructions, block ciphers, AES security. Secret-key encryption, symmetric encryption schemes, one-time pad, Indistinguishability from Random Ciphertext (INDR), Indistinguishability under Chosen Plaintext Attack (IND-CPA), INDR implies IND-CPA, deteministic symmetric encryption schemes.

4.1 How to build PRFs

We began lecture with a brief recap on the contsruction of PRFs using Range Extension, and the associated hybrid argument used to prove this construction.

Now, what are some actual ways for us to build a PRF? There are a few options:

1. Theoretical constructions – this relies on minimal assumptions of a one-way function (functions that are easy one-way, but nearly impossible on the inverse). Typically very inefficient but cool!
2. Direct constructions – proably secure constructions based on concrete "standard" assumptions like cyclic group assumptions
3. Direct construction using block ciphers – this is very much favored in practice, but the down side is that the security expectations are almost entirely heuristic based.

Definition (Block ciphers)**.** A keyed function $E : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$ is a **block cipher** if for all $\lambda \in \mathbb{N}$ and $k \in \{0,1\}^\lambda$, $E(K,\cdot)$ is a *permutation.*

Moreover, both $E(K,\cdot)$ and $E^{-1}(K,\cdot)$ are efficiently computable, given k.

In the community, the AES block cipher is the most widely adopted one, for which we have the *assumption* that: For every $\mathcal{A}$ running t steps and making q evaluations, we have

$$\mathsf{Adv}^{\text{prf}}_{[AES]}(\mathcal{A}, \lambda = 128) \leq \frac{t}{2^{126}} + \frac{q^2}{2^{129}}$$

Again, note here that this is an assumption because the security of block ciphers are entirely heuristics. It's not simply reducted to any hard problem or anything, but overtime this was the best advantage any adversary have goten.

Here, the $\frac{t}{2^{126}}$ term describes the sucess advantage of the best-possible key recovery attack discovered, which gave us a 4x speed up over brute-force key search. Furthermore, $\frac{q^2}{2^{129}}$ is the upperbound on the probability that two outputs of a random function collide.

We're now going to put the conversation regarding PRF constructions aside. Eventually, we will circle back with a discrete log-based PRF contsruction, or other constructions based on hard math problems.

4.2 Intro to secret-key encryption

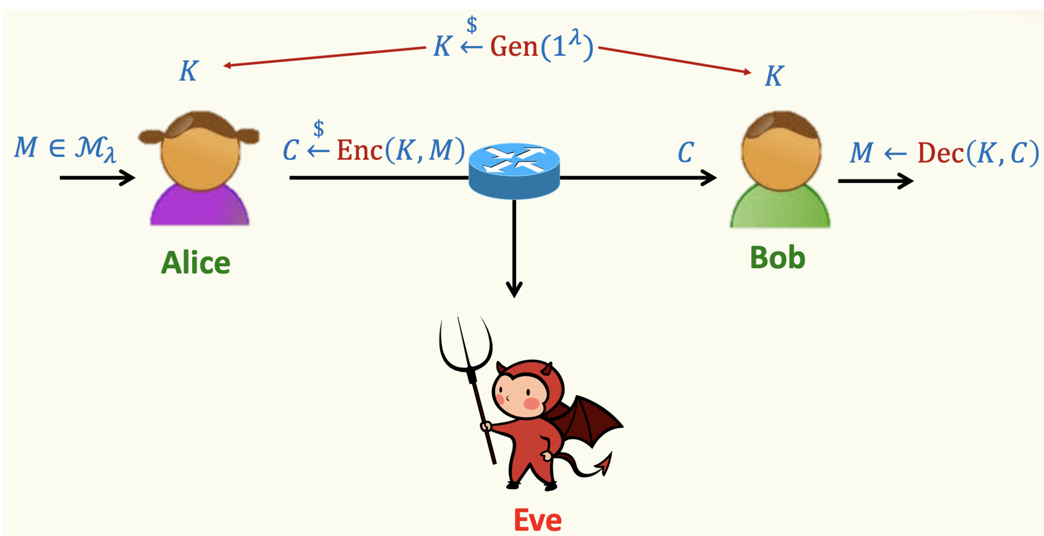
We're now going to look at a completely different security scenario. Here, Alice and Bob share some secret key K, of which Alices uses to encode her plaintext M into some ciphertext C. She then passes on C over some insecure channel to bob, who will decrypt C using K to receive the original message M.

The goal here is that any adversary eve observing C should not learn *anything* about M.

Definition (Secret-key encryption scheme)**.** A **secret-key encyrption scheme** with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is a triple of (efficient) algorithms $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, where:

- The (randomized) **key-generation algorithm** Gen takes the security parameter 1^λ (in unary) as input, and outputs a key K.
- The (psosibly randomized) **encryption algorithm** Enc takes as input a key K output by $\text{Gen}(1^\lambda)$ and a plaintext $M \in \mathcal{M}_\lambda$, and outputs a ciphertext $C \xleftarrow{\$} \text{Enc}(K, M)$.
- The **decryption algorithm** Dec satisfies correct decryption, i.e., for every $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}_\lambda$,

$$\Pr\left[K \xleftarrow{\$} \text{Gen}(1^\lambda) : \text{Dec}(K, \text{Enc}(K, M)) = M\right] = 1$$



Great, now we've discussed the *semantics* and *syntax*. It's time to start thinking about *security*.

Example 4.1. Consider the following "one-time pad" scheme:

Algorithm 15 One-time pad scheme

procedure $\textsc{Gen}(1^\lambda)$
  $K \xleftarrow{\$} \{0,1\}^\lambda$
  return K

procedure $\textsc{Enc}(M)$
  return $M \oplus K$

procedure $\textsc{Dec}(C)$
  return $C \oplus K$

First things first, is this a secret-key encyrption scheme? YES! Because $\mathsf{Dec}(K, \mathsf{Enc}(K, M)) = K \oplus (M \oplus K) = M$ for every $M, K \in \{0,1\}^\lambda$, so it satisfies the correctness criteria.

Now, should we use this scheme? Well, it depends right? If we are passing along just a single text, this scheme is actually perfect! Regardless of our text, because of the XOR, the ciphertext should look uniformly random to the adversary.

But if we were to repeatedly use this scheme, we should see that a "smart-enough" adversary will easily be able to figure out the XOR operation along with the key.

So obviously, security has been thoroughly discussed in the community. Overtime, the semantic security definition that emerged was "The ciphertext leaks nothing about the plaintext, other than what we knew to start with, and the length of the plaintext."

With this came the **Indistinguishability from Random Ciphertext (INDR)** notion. Essentially, every time we encrypt a new chosen plaintext, the ciphertext should "look random" to some bounded adversary without the key.

This means that even if we were to encrypt the same plaintext twice, two different ciphertexts should be outputted by Enc. Turns out, this criteria actually is a bit stronger than what we need to satisfy the above security definition (think about why?).

Algorithm 16 INDR Game

Game $\text{INDR}_{\Pi,b}(\mathcal{A},\lambda)$
  $K \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Enc}_b}(1^\lambda)$
  return $(b' = 1)$

Oracle $\textsc{Enc}_0(M)$
  $C \xleftarrow{\$} \mathsf{Enc}(K, M)$
  return C

Oracle $\textsc{Enc}_1(M)$
  $C \xleftarrow{\$} \{0,1\}^{\text{len}(|M|)}$
  return C'

This gives us the advantage

$$\mathsf{Adv}^{\text{indr}}_{\Pi}(\mathcal{A},\lambda) = |\Pr[\text{INDR}_{\Pi,1}(\mathcal{A},\lambda)] - \Pr[\text{INDR}_{\Pi,0}(\mathcal{A},\lambda)]|$$

So what did we mean when we mentioned that this notion is stronger than what's needed? In the real world, maybe we have a scheme where all ciphertexts are easily distinguishable from random (e.g. leading bit is always 0), but they're all indistinguishable from some message-independent reference distribution. If we have such a scheme, it would actually already satisfy that "ciphertext leaks nothing about the plaintext" criteria as established before.

For example, let Π be some INDR secure symmetric encryption scheme. We now define some scheme Π' where $\Pi'.\mathsf{Enc}' := \Pi.\mathsf{Enc}(K, M)\|0$. Now, Π' should intuitively be "good enough", but it is definitely *not* INDR secure, since it always ends in a 0.

Let's loosen things a bit and introduce **Indistinguishability under Chosen Plaintext Attack (IND-CPA)**.

Algorithm 17 IND-CPA Game

Game $\text{IND-CPA}_{\Pi,b}(\mathcal{A},\lambda)$
  $K \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Enc}_b}(1^\lambda)$
  return $(b' = 1)$

Oracle $\textsc{Enc}_b(M_0, M_1)$
  $C \xleftarrow{\$} \mathsf{Enc}(K, M_b)$
  return C

This is still an indistinguishability game, so we have a familiar advantage definition

$$\mathsf{Adv}_{\Pi}^{\text{ind-cpa}}(\mathcal{A}, \lambda) = |\Pr[\text{IND-CPA}_{\Pi,1}(\mathcal{A}, \lambda)] - \Pr[\text{IND-CPA}_{\Pi,0}(\mathcal{A}, \lambda)]|$$

Okay, we've intuitively thought about how INDR is "stronger" than IND-CPA. Can we formalize this (via a reduction)?

Theorem 4.1.

For every ind-cpa adversary $\mathcal{A}$, there exists an indr adversary $\mathcal{B}$ such that

$$\mathsf{Adv}_{\Pi}^{\textit{ind-cpa}}(\mathcal{A}, \lambda) \leq 2\mathsf{Adv}_{\Pi}^{\textit{indr}}(\mathcal{B}, \lambda)$$

Further, $\mathcal{B}$ is only slightly less efficient than $\mathcal{A}$.

Corollary 4.1.1.

If Π is INDR secure, then it is also IND-CPA secure.

I'm going to sound like a broken record here, but again, since we're trying to prove INDR $\Rightarrow$ IND-CPA, we approach this by contraposition. This means for an adversary $\mathcal{A}$ against IND-CPA, we should be able to reduce it to some adversary $\mathcal{B}$ against INDR.

For the sake of room and time, we are not going to write out the entire proof, but perhaps here's a very quick proof sketch.

Proof. (sketch)

We use hybrid argument. Let G_0 be the IND-CPA$_0$ game and G_2 be the IND-CPA$_2$ game, and let $\mathcal{A}$ be some IND-CPA adversary that interacts with these two games.

From here, define G_1 and G_1' as some intermediate games, where $|\Pr[G_0] - \Pr[G_1]| = \mathsf{Adv}_F^{\text{indr}}(\mathcal{B}_1)$ for some INDR adversary $\mathcal{B}_1$. Same for G_1' and G_2. $\square$

Now let's briefly touch on the notion of "randomness". Say $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is "deterministic", i.e., $\mathsf{Enc}(K, M)$ always outputs the same ciphertext for the same plaintext M. Can Π be IND-CPA secure?

Consider the following adversary:

Adversary $\mathcal{A}^{\text{Enc}_b}(1^\lambda)$
    Pick $M_0 \neq M_1, M_0, M_1 \in \mathcal{M}_\lambda$
    $C_0 \leftarrow \mathsf{Enc}_b(M_0, M_0)$
    $C_1 \leftarrow \mathsf{Enc}_b(M_0, M_1)$
    If $C_0 = C_1$ **then return** 1
    Else return 0

We have the following advantage:

$$\mathsf{Adv}_{\Pi}^{\text{ind-cpa}}(\mathcal{A}, \lambda) = |\Pr[\text{IND-CPA}_{\Pi,0}(\mathcal{A}, \lambda)] - \Pr[\text{IND-CPA}_{\Pi,1}(\mathcal{A}, \lambda)]| = |1 - 0| = 1$$

Why is this intuitively true? When $b = 0$, Enc will encrypt the first argument, which is M_0 in both cases. Since Π is deterministic, we are guaranteed to have $\Pi.\mathsf{Enc}(M_0) = \Pi.\mathsf{Enc}(M_0)$, implying $C_0 = C_1$ and therefore $\mathcal{A}$ will return "1" with probability 1.

However, when $b = 1$, Enc will encrypt the second argument, which is M_0 for the first proceduce call and M_1 for the second. Since Π is a symmetric encryption scheme, $\Pi.\mathsf{Enc}$ must be an invertible operation and therefore one-to-one. This means for $M_0 \neq M_1$, we are guaranteed to have $C_0 = \mathsf{Enc}(M_0) \neq \mathsf{Enc}(M_1) = C_1$, meaning $\mathcal{A}$ will always return "0".

5 Lecture 05: Apr. 13th

summary

5.1 Secret-key encryption from PRFs

We began lecture with a brief overview of the secret-key encryption paradigm. Specifically, recall that secret-key encryption schemes are described by a triple of algorithms (Gen, Enc, Dec) that satisfies the correctness property. We now talk a bit about contsructing secret-key encryption schemes.

Let $F : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ assumed to be a PRF. Our task will be to build $\Pi[F] = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

Algorithm 18 Encryption scheme $\Pi[F]$

procedure $\textsc{Gen}(1^\lambda)$
    $K \xleftarrow{\$} \{0,1\}^\lambda$
    return K

procedure $\textsc{Enc}(K, M)$
    $R \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
    $C \leftarrow (R, F(K,R) \oplus M)$
    return C

procedure $\textsc{Dec}(K, C = (C_1, C_2))$
    return $C_2 \oplus F(K, C_1)$

Theorem 5.1.
For every indr adversary $\mathcal{A}$ encrypting q plaintexts, there exists a prf adversary $\mathcal{B}$ making q queries such that

$$\mathsf{Adv}^{indr}_{\Pi[F]}(\mathcal{A}, \lambda) \leq \mathsf{Adv}^{prf}_{F}(\mathcal{B}, \lambda) + \frac{q^2}{2^{n+1}}$$

Further, $\mathcal{B}$ is only slightly less efficient than $\mathcal{A}$.

Corollary 5.1.1.
If F is a PRF, then $\Pi[F]$ is INDR secure, and hence also IND-CPA secure.

Let's prove this.

Proof. We argue via hybrid argument.

Algorithm 19 Game G_0

Game $G_0()$
  $K \xleftarrow{\$} \{0,1\}^\lambda$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{Enc}}(1^\lambda)$
  return $(b' = 1)$

Oracle Enc(M)
  $R \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $C \leftarrow (R, F(K,R) \oplus M)$
  return C

Claim 0. $\Pr[G_0] = \Pr\left[\mathrm{INDR}_{\Pi[F],0}(\mathcal{A},\lambda)\right]$. This should be immediate upon observation.

Algorithm 20 Game G_1

Game $G_0()$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{Enc}}(1^\lambda)$
  return $(b' = 1)$

Oracle Enc(M)
  $R \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  if $T[R] = \perp$ **then**
    $T[R] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  $C \leftarrow (R, T[R] \oplus M)$
  return C

The difference now between games G_0 and G_1 is that in G_1, rather than using a PRF F, we are directly constructing some random function within **Enc**. Intuitively then, the distinguishing difference between G_0 and G_1 should be the same as the PRF advantage of some adversary $\mathcal{B}$.

Claim 1. There exists adversary $\mathcal{B}$ such that $|\Pr[G_0] - \Pr[G_1]| = \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{B},\lambda)$.

...

The diffenece here is that we set $T[R]$ randomly, even if it was already set to something else before, overwriting its value.

Claim 2. $\Pr[G_3 \text{ sets bad}] \leq \frac{q^2}{2^{n+1}}$. We'll come back to this in a bit.

But notice that in both if-else branches above, we are uniformly randomly sampling $T[R]$. This is bad code quality!! We can reduce G_3 down a bit and say

Algorithm 21 G_3

Game $G_3()$
  bad $\leftarrow$ false
  $b' \xleftarrow{\$} \mathcal{A}^{\mathbf{Enc}}(1^\lambda)$
  return $(b' = 1)$

Oracle Enc()
  $R \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $T[R] \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
  $C \xleftarrow{\$} (R, T[R] \oplus M)$
  return C

"But the bad flag is gone!" – me in lecture. But it actually doesn't matter. The bad flag is just a proof technique for us to justify the differences between G_2 and G_3 (sampling regardless of previous sample or not), which is the probability established by the theorem above.

But we can simplify further by arguing that C is essentially a randomly sampled string.

...

□

Yeah, that entire section was just one proof. lol.

5.2 Modes of operation

Recall our previous conversation about constructing PRFs from block ciphers. Essentially, the PRF F we've used is really just some block cipher $E : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$.

But remember that modern-day block ciphers are AES, which has a messsage space of $n = 128 = 16$ bytes. This ultimately isn't very helpful if we're hoping to encrypt arbitrarily long messages.

We resolve this issue using **modes of operation**, which is to use a block cipher to encrypt arbitrarily long messages. There are two predominant modes: **counter-mode encryption (CTR)** and **cipher-block caining (CBC)**.

...

5.3 Intro to authenticated encryption

So far in the world of secret-key encryption as we know it, we've operated under the assumption that some adversary eve is sitting silently and looking to read and understand our ciphertexts C.

However, in the real world, if messages are being sent over some channel fully controlled by eve, nothing is stopping them to *actively tamper* with the ciphertexts C, even without understanding it.

...

problems in intergity can lead to problems in privacy

...

6 Lecture 06: Apr. 15th

summary

6.1 Integrity Notions

Recall the general goal of integrity: we've previously assumed the adversary is *passive*, observing ciphertexts and trying to understand them. However, towards the end of last lecture, we started considering *active* adversaries, directly changing ciphertexts with the goal of getting them to decrypt to unintentional things.

In this new view, we informally define some integrity security notion: it must be infeasible to produce a new cipheretxt which decrypts to a valid plaintext, even after seeing a large number of valid ciphertexts for the same key. We call this notion *ciphertext integrity*, denoted **INT-CTXT**.

The ultimate goal is to achieve IND-CPA + INT-CTXT, which we call *Authenticated Encryption (AE)*. But before we get there, let's formalize INT-CTXT.

There are actually two notions we want out of INT-CTXT:

- Adversary cannot inject encrypted traffic, not originally sent by Alice
- Confidentiality: adversary cannot learn anything useful by submitting (bogus) ciphertexts to Bob.

Algorithm 22 INT-CTXT Security Game

Game INT-CTXT$_\Pi(\mathcal{A}, \lambda)$
    $K \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
    $\mathcal{Q} \leftarrow \emptyset$
    win $\leftarrow$ false
    Run $\mathcal{A}^{\textbf{Enc, Dec}}(1^\lambda)$ **return** win

Oracle Enc(M)
    $C \xleftarrow{\$} \mathsf{Enc}(K, M)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{C\}$
    return C

Oracle Dec(C)
    if $C \notin \mathcal{Q} \land \mathsf{Dec}(K, C) \neq \perp$ **then**
        win $\leftarrow$ true
        return true
    return false

This gives us the following advantage:

$$\mathsf{Adv}_{\Pi}^{\text{int-ctxt}}(\mathcal{A}, \lambda) = \Pr[\text{INT-CTXT}_{\Pi}(\mathcal{A}, \lambda)]$$

There is a much similar, but intuitively slightly weaker notion of *Plaintext Security* **INT-PTXT**.

Algorithm 23 INT-PTXT Security Game

Game $\text{INT-PTXT}_{\Pi}(\mathcal{A}, \lambda)$
  $K \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
  $\mathcal{M} \leftarrow \emptyset$
  win $\leftarrow$ false
  Run $\mathcal{A}^{\textbf{Enc, Dec}}(1^\lambda)$ **return** win

Oracle Enc(M)
  $C \xleftarrow{\$} \mathsf{Enc}(K, M)$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{M\}$
  return C

Oracle Dec(C)
  if $\mathsf{Dec}(K, C) \notin \mathcal{M} \cup \{\perp\}$ **then**
    win $\leftarrow$ true
    return true
  return false

This gives us the following advantage:

$$\mathsf{Adv}_{\Pi}^{\text{int-ptxt}}(\mathcal{A}, \lambda) = \Pr[\text{INT-PTXT}_{\Pi}(\mathcal{A}, \lambda)]$$

This notion gives us the security where: it must be infeasible to produce a ciphertext that decrypts to a new valid plaintext, even after seeing a large number of ciphertexts for the same key.

...

Theorem 6.1.
For every ind-cca adversary $\mathcal{A}$, there exists an ind-cpa adversary $\mathcal{B}_1$ and int-ctxt adversary $\mathcal{B}_2$ such that

$$\mathsf{Adv}_{\Pi}^{\textit{ind-cca}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}_{\Pi}^{\textit{ind-cpa}}(\mathcal{B}_1, \lambda) + \mathsf{Adv}_{\Pi}^{\textit{int-ctxt}}(\mathcal{B}_2, \lambda)$$

Further, $\mathcal{B}_1$ and $\mathcal{B}_2$ are only slightly less efficient than $\mathcal{A}$.

...

6.2 Generic Composition

Now that we've talked all about the theoretical notions of authenticated encryption, how do we actually build one?

The classical recipe is to use *generic composition*. We combine some IND-CPA secure $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with some UF-CMA secure $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\ell$. One immediate issue is that we don't really know how to build a veriable length MAC, but we will get back to that in a second.

First up, we have: Encrypt-then-MAC.

Algorithm 24 Encrypt-then-MAC

procedure $\textsc{EtM.Gen}(1^\lambda)$
  $K_1 \xleftarrow{\$} \Pi.\mathsf{Gen}(1^\lambda); K_2 \xleftarrow{\$} \{0,1\}^\lambda$
  return (K_1, K_2)

procedure $\textsc{EtM.Enc}((K_1, K_2), M)$
  $C' \xleftarrow{\$} \Pi.\mathsf{Enc}(K_1, M)$
  $T \leftarrow F(K_2, C')$
  return $C = (C', T)$

procedure $\textsc{EtM.Dec}((K_1, K_2), (C', T))$
  if $T = F(K_2, C')$ **then**
    $M \leftarrow \Pi.\mathsf{Dec}(K_1, C')$
    return M
  else
    return $\perp$

As you can see, we literally just slapped a mac together with a ind-cpa Π. Let's establish the security notions now.

Theorem 6.2.
For every int-ctxt adversary $\mathcal{A}$, there exists a uf-cma adversary $\mathcal{B}$ s.t.

$$\mathsf{Adv}_{EtM}^{int\text{-}ctxt}(\mathcal{A}, \lambda) \le \mathsf{Adv}_{F}^{uf\text{-}cma}(\mathcal{B}, \lambda)$$

Further, $\mathcal{B}$ is only slightly less efficient that $\mathcal{A}$.

Theorem 6.3.
For every ind-cpa adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that

$$\mathsf{Adv}_{EtM}^{ind\text{-}cpa}(\mathcal{A}, \lambda) \le \mathsf{Adv}_{\Pi}^{ind\text{-}cpa}(\mathcal{B}, \lambda)$$

Further, $\mathcal{B}$ is only less efficient than $\mathcal{A}$.

But now a thought experiment, can EtM be INDR secure if Π is INDR secure? NO. The output of EtM is some pair (C', T), and we have no guarantee that the tag T is indistinguishable from random, and therefore, even if C' is indistinguishable from random, our overall ciphertext C may not be.

Okay, now since there is an Encrypt-then-MAC, there must be a MAC-then-Encrypt right? There is.

Algorithm 25 MAC-then-Encrypt

procedure $\text{MtE.Gen}(1^\lambda)$
  $K_1 \xleftarrow{\$} \Pi.\mathsf{Gen}(1^\lambda); K_2 \xleftarrow{\$} \{0,1\}^\lambda$
  return (K_1, K_2)

procedure $\text{MtE.Enc}((K_1, K_2), M)$
  $T \leftarrow F(K_2, M)$
  $C \xleftarrow{\$} \Pi.\mathsf{Enc}(K_1, (M, T))$
  return C

procedure $\text{MtE.Dec}((K_1, K_2), C)$
  $M' \leftarrow \mathsf{Dec}(K_1, C)$
  if $M' \neq \perp$ **then**
    Parse M' as (M, T)
    If $F(K_2, M) = T$ **then return** M
  return $\perp$

6.3 VIL-PRFs

Let's now zone into how to bulid variable length MACs. We want to build some PRF $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\ell$, but AES only gives us block ciphers where $E : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$. How do we extend this domain to take variable length?

But before we get to that, we'd need some new definitions and tools.

Definition (almost universal hashing)**.** A keyed function $H : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^n$ is **δ-almost-universal** (**δ-AU**) if for all $M \neq M'$ we have

$$\Pr\Big[K \xleftarrow{\$} \{0,1\}^\lambda : H(K, M) = H(K, M')\Big] \leq \delta(\max\{|M|, |M'|\})$$

...

7 Lecture 07: Apr. 20th

summary

7.1 Finite groups

We began lecture with a quick callback to everything we've covered thus far in the course. We've explored lots of cryptographic primitives and their associated properties, but we've mostly taken their construction for granted.

In the next few lectures, we will discuss some mathematics used to back these cryptographic primitives based on their "hardness".

But before that, a quick (re)fresher on group theory.

Definition (Finite groups)**.** A (finite) **group** $(\mathbb{G}; *)$ is a (finite) set $\mathbb{G}$ and a binary operation $*$ which satisfy:

1. Closure: $\forall a, b \in \mathbb{G}; a * b \in \mathbb{G}$
2. Identity: $\exists 1 \in \mathbb{G}$ such that $\forall a \in \mathbb{G} : 1 * a = a * 1 = a$
3. Associativity: $\forall a, b, c \in \mathbb{G}, (a * b) * c = a * (b * c)$
4. Inerse: $\forall a \in \mathbb{G} : \exists b \in \mathbb{G}$ such that $a * b = b * a = 1$. We denote b as the *inverse* of a, that is $b = a^{-1}$.

The size of the group, denoted $|\mathbb{G}|$, is called the *order* of $\mathbb{G}$.

Before we get into some examples, let's review modular arithmetics. We begin with

Theorem 7.1 (Division theorem).
For any a, N with $N > 0$, there exists a unique q and r such that $a = Nq+r$ and $0 \leq r < N$.

We define $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$. Also defore **modular addition** an $+_N : \mathbb{Z}_N \times \mathbb{Z}_N \to \mathbb{Z}_N$ such that

$$a +_N b = (a + b) \bmod N$$

Fact. $(\mathbb{Z}_N; +_N)$ is a group.

We define **modular multiplication** as $\times_N : \mathbb{Z}_N \times \mathbb{Z}_N \to \mathbb{Z}_N$ such that

$$a \times_N b = (a \cdot b) \bmod N$$

Fact. $(\mathbb{Z}_N; \times_N)$ is NOT a group. This is because some elements don't have inverses. Let's explore inverses:

Theorem 7.2 (Bezout's lemma).
There exists $b, k \in \mathbb{Z}$ such that $ab + kN = 1$ if and only if $\gcd(a, N) = 1$. Furthermore, b, k can be found efficiently via the Extended Euclidean Algorithm for computing GCDs.

We now define a set where , for every $N \geq 1$, define

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : \gcd(a, N) = 1\}$$

Fact. $(\mathbb{Z}_N^*, \times_N)$ is a group, and it's order is denoted by the Euler totient function $\varphi(N)$.

Before we dive in further, we discuss some notational things. First, when $*$ is understood, we write $\mathbb{G}$ instead of $(\mathbb{G}; *)$. Additionally, we will be using multiplicative notation for group operation.

So remember why we even started talking about groups in the first place: we're on a quest seeking out "hard" mathematical operations to back our cryptographic primitives.

When compuing over $\mathbb{Z}_N/\mathbb{Z}_N^*$, operations are efficient if they can be computed in worst-case time $O((\log N)^2)$. Almost everything we've talked about are efficient. Let's turn and look at exponentiation.

Theorem 7.3 (Euler's theorem).
If $\mathbb{G}$ is finite, then $x^{|\mathbb{G}|} = 1$ for all $x \in \mathbb{G}$.

Definition (Generator & Cyclic groups)**.** g is a **generator** of $\mathbb{G}$ if

$$\langle g \rangle = \{g^0 = 1, g^1, \ldots, g^{\mathbb{G}-1}\} = \mathbb{G}$$

A group is said to be **cyclic** if it contains a generator.

So are $\mathbb{Z}_N$ always cyclic? Yes! All additive groups are cyclic since $\langle 1 \rangle = \{1, \ldots, N-1\}$.

So are $\mathbb{Z}_N^*$ always cyclic? Not necessarily.

Theorem 7.4 (Primitive roots theorem).
The group $\mathbb{Z}_N^*$ is cyclic if and only if $N = 2, 4, p^k, 2p^k$ for some $k \in \mathbb{Z}$ and prime $p > 2$.

So like what's the point of all this stuff?

7.2 The Discrete Logarithm problem

Let $(\mathbb{G}; *)$ be a group. Goal: Given $x \in \mathbb{G}, e \in \mathbb{N}, e \geq 0$, compute $x^e \in \mathbb{G}$. Is this a *hard* problem?

Well, it looks like we'd have to multiply a number exactly e times, and depending on the value of e, this may break that $O((\log N)^2)$ barrier we discussed earlier! Yay!

JK. If we do exponentiation using a clever algorithm (Square-and-multiply), we can actually cut it down to just $O(\log_2 e)$ group operations. So this is not great. We need something harder.

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order N. We've seen that $x \mapsto g^x$ is the exponentiation operation – easy. Let's now look at the converse: $g^x \mapsto x$, also known as the **discrete logarithm** operation.

Spoiler alert! Discrete log is hard. Let's formalize this:

Definition (Discrete log assumption)**.** For a cyclig group $\mathbb{G} = \langle g \rangle$ and an adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}) = \Pr\left[x \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|} : \mathcal{A}(g^x) = x\right]$$

We say that the **DL problem is hard** if $\mathsf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A})$ is negligible for all PPT $\mathcal{A}$.

We model groups parametrized by security parameter $\lambda \in \mathbb{N}$.

Definition (Grou pparameter generator)**.** A **Group parameter generator** `GGen` is a (probabilistic) polynomial time algorithm which, on input 1^λ, outputs

- The description of a group $\mathbb{G}$
- A generator g of $\mathbb{G}$
- The order $m = |\mathbb{G}|$, which satisfies $2^\lambda \leq m \leq 2^{\lambda+1}$

Further, there exists an algorithm to compute group operations in $\mathbb{G}$ in time polynomial to λ, and its elements have poly size λ.

It's not a crypto lecture notes without some good-old pseudocode:

Algorithm 26 Discrete Log Assumption

```
Game DL_GGEN(A, λ)
    (G, g, m) <-$ GGen(1^λ)
    x <-$ Z_m
    x' <-$ A(G, g, g^x)
    return (x = x')
```

And we define the advantage as

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathrm{GGen}}(\mathcal{A}, \lambda) = \Pr[\mathsf{DL}_{\mathrm{GGen}}(\mathcal{A}, \lambda)]$$

Logically, we now want to pick a group $\mathbb{G}$ where DL is, like, very very hard. Let $\mathbb{G} = \langle g \rangle$ be cyclic, order $m = |\mathbb{G}| \approx 2^\lambda$. Naively, this will take $O(m) = O(2^\lambda)$ group operations. Can we do better?

YES! Much like with SquareMult and exponentiation, there is a clever algorithm (Baby-Step, Giant-Step) that can be extended and implemented in $O(\sqrt{q})$ time and memory, where q is the largest prime dividing $m = |\mathbb{G}|$.

And thus, "good" groups should have order m with large prime factors. In most cases, we pick m to be prime. Note that this is *not* the same as $\mathbb{Z}_p^*$! There exists clever algorithms that solves DLog in $\mathbb{Z}_p^*$ with efficient running time.

Goal: want a different approach to build cryptography-friendly groups, for which (1) DL is assumed to be true, and (2) best known attack is BSGS (so it's predictable). We do this with *elliptic curves* (but won't discuss it for now).

As a summary for today:

- Algebra & Groups: want to leverage "hard problems" to instantiate primitives
- The discrete logarithm problem: baseline "hard" computational problem in finite groups
- **Next time:** Diffie-Hellman assumption and its applications

8 Lecture 08: Apr. 22nd

summary

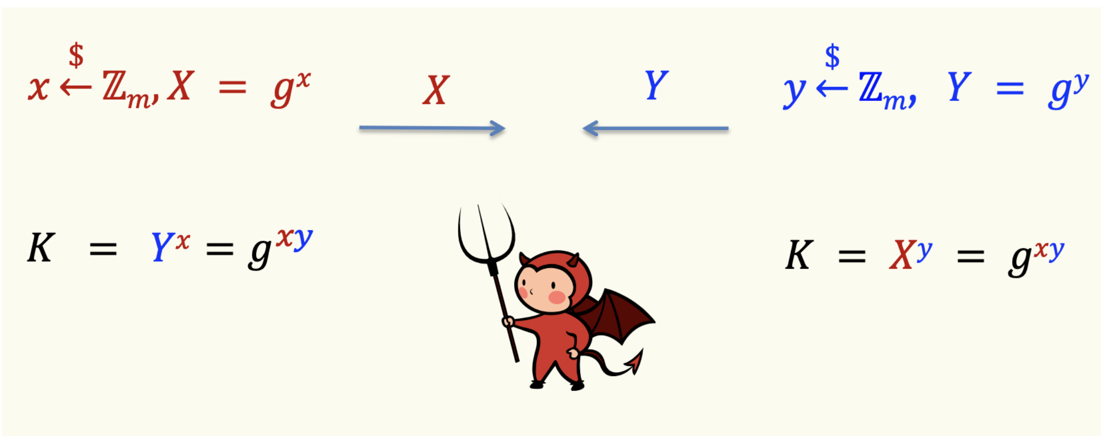
8.1 DDH and Diffie-Hellman

We began lecture with a quick review of Monday's lecture. We saw the discrete logarithm operation, which is essentially the inversion of a group exponentiation. We assumed that this is a mathematically and computationally hard operation.

We saw some "clever" attacks like Baby-Step Giant-Step, but none was clever enough to break the exponential time barrier in the general case.

Okay, back to cryptography. Thus far in the course, we've always assumed that Alice and Bob have just started with this "shared" secret key, and can use this key in authenticated encryption. But do they exacly agree on a secret-key securely, especially over some channel already compromised by Eve?

One such way is known as the **Diffie-Hellman Key Exchange**.



Alice and bob each sample a random exponent $x \xleftarrow{\$} \mathbb{Z}_m$ and $y \xleftarrow{\$} \mathbb{Z}_m$. Then, they each raise some generator g to that power. They pass the result $X = g^x$ and $Y = g^y$ to each other. Finally, they arrive at the same shared key $K = X^y = Y^x = g^{xy}$.

So, what can the adversary do here? Well, since eve only sees X and Y, ...

Let's talk about the security notion we want. Specifically, we assume for now that the adversary is *passive*, meaning it can only observe X and Y, and try to piece together K. It cannot inject new values into the stream of communication between alice and bob.

Since our adversary is passive, we bring back good old distinguishing advantages. The security property we want is that "even in the worst case if K is leaked to the adversary, it should not be able to distinguish between the real key K and some $K' \xleftarrow{\$} \mathbb{G}$."

(X, Y, K) is computationally indistinguishable from (X, Y, K').

Algorithm 27 Decisional Diffie-Hellman (DDH) Assumptionn

```
Game DDH_{GGen,0}(A, λ)
    (G, g, m) ←$ GGen(1^λ)
    x, y ←$ Z_m
    b' ←$ A(G, g, g^x, g^y, g^{xy})
    return (b' = 1)

Game DDH_{GGen,1}(A, λ)
    (G, g, m) ←$ GGen(1^λ)
    x, y, z ←$ Z_m
    b' ←$ A(G, g, g^x, g^y, g^z)
    return (b' = 1)
```

We define the advantage to be

$$\mathsf{Adv}^{\text{ddh}}_{\mathsf{GGen}}(\mathcal{A}, \lambda) = |\Pr[\text{DDH}_{\mathsf{GGen},0}(\mathcal{A}, \lambda)] - \Pr[\text{DDH}_{\mathsf{GGen},1}(\mathcal{A}, \lambda)]$$

Definition (DDH Assumption)**.** The Decisional Diffie-Hellman (DDH) Assumption holds for GGen if $\mathsf{Adv}^{\text{ddh}}_{\mathsf{GGen}}(\mathcal{A}, \lambda)$ is negligible for all PPT $\mathcal{A}$.

Theorem 8.1.

If the DDH assumption holds for GGen, then DL assumption also holds for GGen.

Concretely, for all dl adversary $\mathcal{A}$, there exists a ddh adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{dl}_{GGen}(\mathcal{A}) \leq \mathsf{Adv}^{ddh}_{GGen}(\mathcal{B}) + \frac{1}{2^\lambda}$$

Proof. (sketch)

Consider the following adversary:

Algorithm 28 Adversary against DDH

```
Adversary B(G, g, X, Y, Z)
    x' ←$ A(G, g, X)
    return (Z = Y^{x'})
```

Fact 1. $\Pr[\text{DDH}_{\mathsf{GGen},0}(\mathcal{B}, \lambda)] = \Pr[\text{DL}_{\mathsf{GGen}}(\mathcal{A}, \lambda)] = \mathsf{Adv}^{\text{dl}}_{\mathsf{GGen}}(\mathcal{A}, \lambda)$.

Fact 2. $\Pr[\text{DDH}_{\mathsf{GGen},1}(\mathcal{B}, \lambda)] = \Pr\left[g^z = Y^{x'}\right] = 1/|\mathbb{G}| \leq 1/2^\lambda$

Both facts together would imply

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GGen}}(\mathcal{B}, \lambda) \geq \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GGen}}(\mathcal{A}, \lambda) - \frac{1}{2^\lambda}$$

□

For the converse, we do not believe it to be true. In other words, one can show that there are groups for wich the DL assumption is likely true, but the DDH assumption is certainly false.

8.2 The Naor-Reingold Construction

In prior lectures, we've just *assumed* that a PRF is available. We've seen PRF security with the AES block cipher, which again, the security is *conjectured for* (emperically).

This section, we'll try to construct based on a much stronger, mathematical assumption.

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order $|\mathbb{G}| = m$. Then, we define the key function $\mathrm{NR}^n_{\mathbb{G},g} : \mathbb{Z}_m^{n+1} \times \{0,1\}^n \to \mathbb{G}$, where

$$\mathrm{NR}^n_{\mathbb{G},g}\left((s_\epsilon, k_1, \ldots, k_n), x = x_1 x_2 \ldots x_n\right) = g^{s_\epsilon \prod_{i=1}^n k_1 x_i}$$

> **Intuition.** *We multiply the exponent of g by k_1 whenever $x_1 = 1$, otherwise we do not.*

Theorem 8.2.
For every GGen and $n = n(\lambda)$ and every prf adversary $\mathcal{A}$ making at most $q = q(\lambda)$ queries, there exists a ddh adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{prf}_{NR^n_{GGen}}(\mathcal{A}, \lambda) = q(\lambda) \cdot n(\lambda) \cdot \mathsf{Adv}^{ddh}_{GGen}(\mathcal{B}, \lambda)$$

Before we dive into this proof, let's establish some tools first. We introduce ℓ-DDH:

Lemma 8.3.
For every $\ell = \ell(\lambda)$ and GGen, and every l-ddh adversary $\mathcal{A}$, there exists a ddh adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\ell\text{-}ddh}_{GGen}(\mathcal{A}, \lambda) = \ell(\lambda) \cdot \mathsf{Adv}^{ddh}_{GGen}(\mathcal{B}, \lambda)$$

Further, $\mathcal{B}$ is only slightly less efficient than $\mathcal{A}$.

Let's prove this lemma first.

Proof. Consider the following hypbrid argument, where we define exactly ℓ games:

Algorithm 29 ℓ-DDH Hybrid Argument

```
Game G_i(): // i ∈ {0, 1, ..., ℓ}
    (𝔾, g, m) <-$ GGen(1^λ)
    x, y_1, ..., y_ℓ <-$ ℤ_m
    for j = 1 to n do
        If j ≤ i Then z_j <-$ ℤ_m Else z_j ← x y_i
    b' <-$ 𝒜(𝔾, g, g^x, g^{y_1}, ..., g^{y_ℓ}, g^{z_1}, ..., g^{z_ℓ})
    return (b' = 1)
```

Claim. For every $i \in \{0, 1, \ldots, \ell-1\}$, there exists an adversary $\mathcal{B}_i$ such that $\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GGen}}(\mathcal{B}_i, \lambda) = |\Pr[G_i] - \Pr[G_{i+1}]|$

Using this claim (which we will show later), we see that

$$\begin{aligned}
\mathsf{Adv}^{\ell\text{-ddh}}_{\mathsf{GGen}}(\mathcal{A}, \lambda) &= |\Pr[G_0] - \Pr[G_\ell]| \\
&= \left| \sum_{i=0}^{\ell-1} \Pr[G_i] - \Pr[G_{i+1}] \right| \\
&\le \sum_{i=0}^{\ell-1} |\Pr[G_i] - \Pr[G_{i+1}]| \\
&= \sum_{i=0}^{\ell-1} \mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GGen}}(\mathcal{B}_i, \lambda)
\end{aligned}$$

which is exactly the desired result.

...

But is this enough?

...

□

As a summary of today's lecture:

- DDH and Diffie-Hellman Key Agreement: passive security is equivalent to DDH assumption, which is stronger than DL assumption
- The Naor-Reingold PRF construction: complex hybrid argument through polynomially many hybrids

9 Lecture 09: Apr. 27th

summary

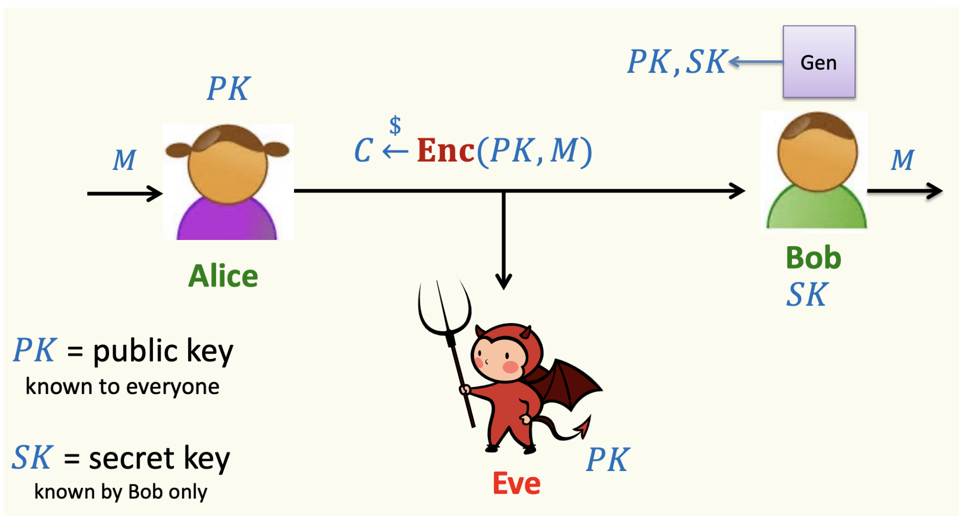
9.1 Public key encryption

We began lecture with a very quick review of the Decisional Diffie-Hellman (DDH) assumption. Typically, we try to perform DDH over groups with prime order. That means, if we try to utilize $\mathbb{Z}_p^*$, where $|\mathbb{Z}_p^*| = p - 1$, it's not great.

In homework 2, we will consider $\mathbb{Z}_p^*$ where $p = 2q + 1$ for some prime q. Here, there will exist some $g \in \mathbb{Z}_p^*$ such that $\langle g \rangle$ is a sub-group of order q, which is good. But even then, the best algorithms still run in time sub-exponential in $\lambda = \log p$. This is actually why we're no longer going over RSA encryption in this class, because prime selection simply require p to be huge, and it's slowly becoming less and less practical over the years.

Otherwise, we can also think about **elliptic curves**, sampling points (x, y) from some finite field $\mathbb{F}_p^2$.

We now shift our focus to public-key encryption. We've spoken about key-exchange already – for parties alice and bob to agree on a secret key K, which can then be used for symmetric encryption. But now, we kind of combine these two steps into the same structure.



Bob runs some key generation Gen, which returns a public-key PK and secret-key SK. Anyone on the internet can have access to PK in order to encrypt messages to bob, but the only way to decrypt these messages is with bob's SK.

Definition (Public-key encryption scheme)**.** A **public-key encyrption scheme** with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a triple of (efficient) algorithms $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, where:

- The (randomized) **key-generation algorithm** Gen takes the security parameter 1^λ (in unary) as input, and outputs a public-key/secret-key pair (PK, SK).
- The (psosibly randomized) **encryption algorithm** Enc takes as input the public key PK and a plaintext $M \in \mathcal{M}_\lambda$, and outputs a ciphertext $C \xleftarrow{\$} \text{Enc}(PK, M)$.
- The **decryption algorithm** Dec satisfies correct decryption, i.e., for every $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}_\lambda$,

$$\Pr\left[(PK, SK) \xleftarrow{\$} \text{Gen}(1^\lambda) : \text{Dec}(SK, \text{Enc}(PK, M)) = M\right] = 1$$

But before we get further, why are we using public-key encryption over key exchange? Well, in public-key encryption, bob publishes his PK, such that anyone in the world can encrypt messages to him. This eliminates a lot of overhead, since we wouldn't require bob to complete a key-exchange with every single person he wishes to communicate with.

Alright, let's now talk about construction. We actually don't need to do too much more work, because we can base our contsruction based directly on Diffie-Hellman!

1. Bob samples $x \xleftarrow{\$} \mathbb{Z}_p$ and generates $X \leftarrow g^x$. He publishes X.
2. Alice (who wishes to communicate with bob) also samples $y \xleftarrow{\$} \mathbb{Z}_p$ and generates $Y \leftarrow g^y$. Since Alice has access to bob's public X, by diffie-hellman, she can find $K = X^y = g^{xy}$.
3. Alice can now use her favorite symmetric encryption scheme to get $Z \xleftarrow{\$} \text{SKEnc}(K, M)$ for any message M. She then passes (Y, Z) to bob.
4. Finally, Bob finds the key $K = Y^x = g^{xy}$ and uses the same symmetric encryption scheme to find $M \leftarrow \text{KSDec}(X, Z)$.

This quite literally combined the (1) key-exchange and (2) symmetric encryption together into one scheme, just like we talked about. Does it work?

Intuitively, yes. But it would only work to encrypt a single message, which actually isn't a bad thing. Notice that we're generating a new key for the Symmetric Encryption every time, since alice samples a new y every time.

This means, for our underlying symmetric encyrption scheme, it only needs to be *single message secure*. We can use the intuition from one-time pads to think about

$$\text{SKEnc}(K, M) = K \cdot M \qquad \text{SKDec}(K, Z) = K^{-1} \cdot Z$$

This leads us to the **ElGamal Encryption** scheme. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order $|\mathbb{G}| = m = p$.

Algorithm 30 ElGamal Encryption Scheme

procedure $\textsc{Gen}(1^\lambda)$
  $SK \xleftarrow{\$} \mathbb{G}$
  $PK \leftarrow g^{SK}$
  return (PK, SK)

procedure $\textsc{Enc}(PK, M)$
  $r \xleftarrow{\$} \mathbb{Z}_p$
  return $C = (g^r, M \cdot PK^r)$

procedure $\textsc{Dec}(SK, (C_1, C_2))$
  return $C_2 \cdot C_1^{-SK}$

Naturally, we now argue the security of this scheme. But first, what does security even look like for public-key encryption schemes?

We can still use the intuition from IND-CPA, where we can now define

Algorithm 31 IND-CPA Game (PKE Edition)

Game $\text{IND-CPA}_{\Pi,b}(\mathcal{A}, \lambda)$
  $(PK, SK) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
  $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Enc}_b}(PK)$
  return $(b' = 1)$

Oracle $\textsc{Enc}_b(M_0, M_1)$
  $C \xleftarrow{\$} \mathsf{Enc}(PK, M_b)$
  return C

But beyond just changing a secret key to a public key, remember that we also only need our scheme to be secure *once*. Thus, it is enough to prove IND-CPA security against adversaries that query Enc_b once. But this is not true for symmetric encryption!!

This leads us to believe that single-query IND-CPA should imply the security of multi-query IND-CPA, within public-key encryption.

Theorem 9.1.
For every ind-cpa adversary $\mathcal{A}$ making $q = q(\lambda)$ queries to Enc_b, there exists an ind-cpa adversary $\mathcal{B}$ making a single query to Enc_b such that

$$\mathsf{Adv}_{\Pi}^{\textit{ind-cpa}}(\mathcal{A}, \lambda) = q(\lambda) \cdot \mathsf{Adv}_{\Pi}^{\textit{ind-cpa}}(\mathcal{B}, \lambda)$$

Further, $\mathcal{B}$ is only slightly less efficient than $\mathcal{A}$.

Okay, let's now show security of ElGamal:

Theorem 9.2.

For every ind-cpa adversary $\mathcal{A}$, making a single encryption query, there exists a ddh adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\textit{ind-cpa}}_{\textit{ElGamal}}(\mathcal{A}, \lambda) \leq \mathsf{Adv}^{\textit{ddh}}_{\mathbb{G},g}(\mathcal{B}_1, \lambda) + \mathsf{Adv}^{\textit{ddh}}_{\mathbb{G},g}(\mathcal{B}_2, \lambda)$$

Further, $\mathcal{B}_1$ and B_2 are only slightly less efficient than $\mathcal{A}$.

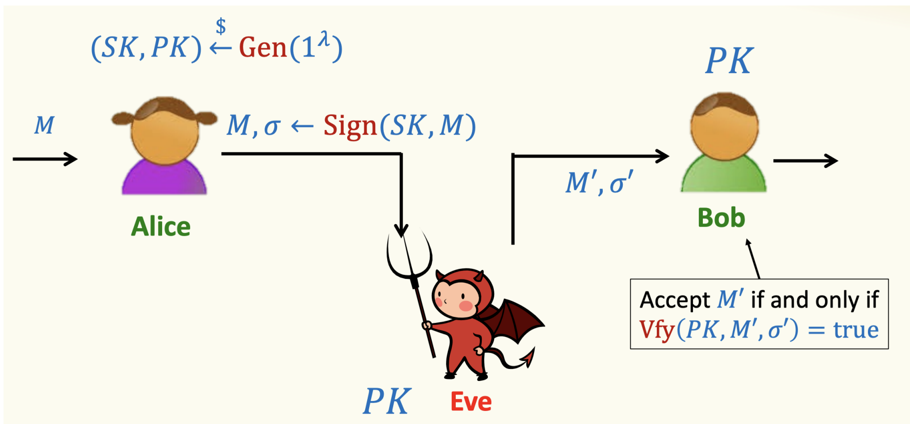
10 Lecture 11: May 4th

summary

10.1 Digital Signature

Oopsies I skipped the last lecture. Err I may or may not catch up the notes.

Anyway, today we're going to be talking about digital signature schemes! See below for the scheme:



The security game is defined as:

Algorithm 32 UF-CMA Game for signatures

1: $(SK, PK) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
2: $Q \leftarrow \emptyset$
3: $(M^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathbf{S}}(PK)$
4: **return** $(M^* \notin Q \wedge \mathbf{Vfy}(PK, M^*, \sigma^*) = \text{true}$

Now that we've constructed semantics, syntax, and security, let's try to build this thing!

Today, we'll talk about a folklore construction called the **full-domain hash** (FDH). It's typically instantiated from RSA, but we'll look at it mostly generically today.

This construction we call a "deterministic public-key encryption", or formally known as **Trapdoor permutations**. The starting point is some PKE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ where

1. Enc is deterministic
2. The ciphertext space is *dense*. For every $C \in \mathcal{C}$, we have $\mathsf{Enc}(PK, \mathsf{Dec}(SK, C)) = C$. This directly implies a bijection $\mathcal{M} \to \mathcal{C}$ for every PK

This is a "trapdoor" because it's easy to go from $\mathcal{M}$ to $\mathcal{C}$, but the converse requires us to break the encryption scheme. It's also a "permutation", because there is a bijection for each (PK, SK) pair.

> **Intuition.** *In other words, the key-gen algorithm* Gen *can be thought of as sampling a permutation between* $\mathcal{M}$ *and* $\mathcal{C}$.

But recall that deterministic PKE's *cannot* be IND-CPA secure. We need another property to take advantage of this "trapdoor-ness" of the scheme.

Algorithm 33 One-Wayness of PKE

Game $\mathrm{OW}_\Pi(\mathcal{A}, \lambda)$
  $(PK, SK) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
  $M \xleftarrow{\$} \mathcal{M}_\lambda$
  $M' \xleftarrow{\$} \mathcal{A}(PK, \mathsf{Enc}(PK, M))$
  return $(M' = M)$

The advantage is defined as

$$\mathsf{Adv}_\Pi^{\mathrm{OW}}(\mathcal{A}, \lambda) = \Pr[\mathrm{OW}_\Pi(\mathcal{A}, \lambda)]$$

Definition (One-wayness). Π is one-way if $\mathsf{Adv}_\Pi^{\mathrm{OW}}(\mathcal{A}, \lambda)$ is negligible for all PPT $\mathcal{A}$

Okay, new property unlocked. Let's go back to talking about a FDH construction. Let $H : \{0,1\}^* \to \mathcal{C}$ be a hash function, and $\Pi = (\Pi.\mathsf{Gen}, \Pi.\mathsf{Enc}, \pi.\mathsf{Dec})$ be a deterministic PKE with a dense ciphertext space.

We then define a signature scheme Σ, where

- $\Sigma.\mathsf{Gen} = \Pi.\mathsf{Gen}$
- $\Sigma.\mathsf{Sign}(SK, M) = \Pi.\mathsf{Dec}(SK, H(M))$
- $\Sigma.\mathsf{Vfy}(PK, M, \sigma) = (H(M) == \Pi.\mathsf{Enc}(PK, \sigma))$

Why does this work intuitively? Well, in order to forge a signature for M^*, attakcer needs to produce $\sigma = \Pi.\mathsf{Dec}(SK, H(M^*))$, i.e., it needs to invert $H(M^*)$.

If we believe in the "one-wayness" of Π, then it shouldn't be possible for an adversary to sign simply by seeing other signatures, because that would require them to go the *other way* – to figure out the permutation by only seeing samples from $\mathcal{C}$.

Our task is now hopefully clear. Let's formalize this via the following theorem:

Theorem 10.1.

For every uf-cma ROM adversary $\mathcal{A}$ making q RO queries, signature queries and $Q-S$ signature. queries, there exists a ow adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{uf\text{-}cma}_{\Sigma}(\mathcal{A}) \leq (q+q_S)\cdot \mathsf{Adv}^{ow}_{\Pi}(\mathcal{B})$$

Further, $\mathcal{B}$ is only slightly less efficient than $\mathcal{A}$.

Proof! It's facts.

10.2 Authenticated Key Exchange

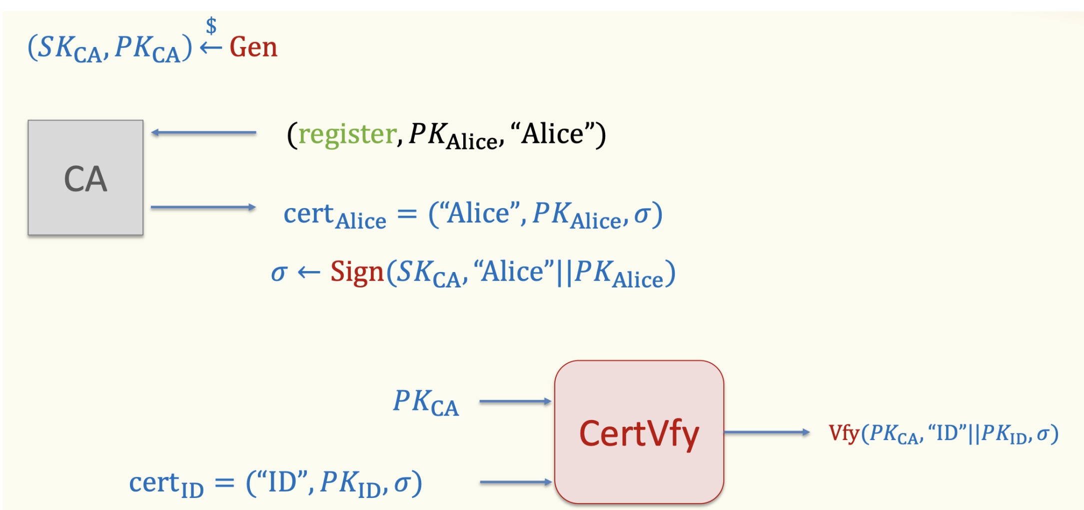
We take a detour and talk about an application of these digital signatures. Prior to today, we consider security of key exchange against passive attackers, where they can only eavesdrop the communication.

Today, we looked at active adversaries – adversaries that can inject / modify messages. In the secret-key environment, we talked about authenticated encryption, a mac-on-ske kinda thing that guarantees security.

But unfortunately, this is not possible in the public-key environment. It is acutally *impossible* to have parties agree on a key in this setting without the reliance on "something trusted".

Just think about it, since the adversary controls the entire channel of communication, how could alice and bob (1) agree on a key, and (2) know there wasn't someone in the middle?

Because of this, we talk about **Certificate Authorities** (CA): some trusted third-party that assigns "certifitaes" to users that binds their identity with their public key.



Ew. Trusting on a third party. Oh well what can we do about it.

With this in mind, let's talk about how we can establish authenticated key exchange. Say Alice (user) is trying to exchange keys with Bob (bank). Bob needs to have some certification (cus its a bank), but not necessarily Alice.

This is how we form *assymetric key exchange*. Consider the following procedure:

1. Bob sends $R \xleftarrow{\$} \{0,1\}^r$ and $\text{cert}_{\text{bank}}$.
2. Alice verifies the certificate to esure shes really talking with Bob, and not some person-in-the-middle.
3. They then perform a regular PKE scheme.

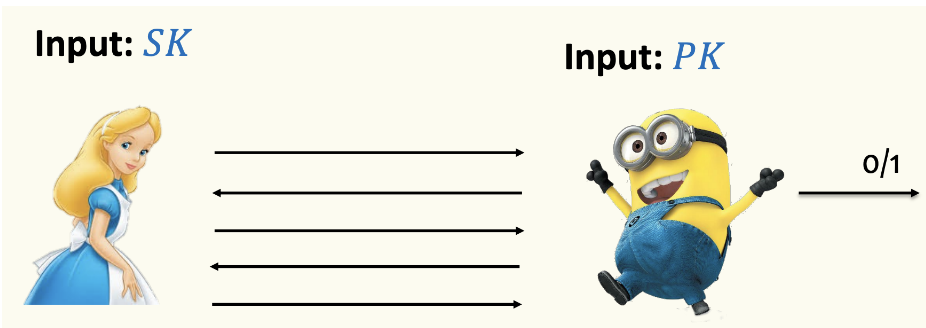
11 Lecture 12: May 6th

summary

11.1 Identification Protocols

We began lecture with a new protocol in identification. Assume we generate some $(PK, SK) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$. We want a protocol where anyone with a PK can "identify" that they're speaking with someone holding SK, and not an impersonator.

Specifically, if alice held SK and bob holds her PK. The two parties should be able to communicate with each other, where at some point bob can output a 0/1 depending on if he thinks hes communicating with real alice or some impersonator.



Importantly, we should have that

$$\Pr[\text{"1" when speaking to alice }] = 1$$

and

$$\Pr[\text{"1" when speaking to impersonator}] = \text{negl}$$

We won't dive super deep into what makes an identification scheme secure in this class as it gets quite nuance, but we will take that for granted for future proofs.

11.2 Katz-Wang Signatures

We now build a signature scheme on top of the **Katz-Wang Identification Protocol**, which goes as follows:

1. We work over a finite cyclic group $\mathbb{G} = \langle g \rangle$ of order p
2. Alice samples $SK = (h, x)$ and $PK = (h, X = g^x, Y = h^x)$ for $h \xleftarrow{\$} \mathbb{G}$ and $x \xleftarrow{\$} \mathbb{Z}_p$
3. At the beginning of the protocol, she samples some $r \xleftarrow{\$} \mathbb{Z}_p$, and sends $A = g^r, B = h^r$ to Bob.
4. Bob samples a challenge $c \xleftarrow{\$} \mathbb{Z}_p$ and sends it back to Alice

5. Alice computes some $z = r + cx$ and sends it to Bob
6. Bob outputs "1" iff $X^c A = g^z$ and $Y^c B = h^z$. Otherwise, he outputs "0".

As for correctness, we see that

$$X^c A = g^{cx} g^r = g^{r+cx} = g^z$$
$$Y^c B = h^{cx} h^r = h^{r+cx} = h^z$$

There IS a reason why we're doing this twice with g and h. Trust it's coming. ...

*****

Now that we know it's secure, let's try to construct a signature scheme from it. First, signature schemes are non-interactive, so we need to make this identification protocol also non-interactive.

To do this, rather than bob sampling a challenge $c \xleftarrow{\$} \mathbb{Z}_p$, we define a hash function (modeled like a random oracle) $H : \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p$. Then,

1. Alice computes $A \leftarrow g^r$ and $B \leftarrow h^r$ as usual
2. Alice also computes $c = H(A, B)$
3. Alice sends $z = r + cx$ to Bob
4. Prior to correction checks, bob first verifies $c = H(A, B)$

Another property of signature schemes that we don't yet have is *message dependence*. We'd want the output of alice to be dependent on the message she is signing. This is a simple addendum – we just hash M along with A, B when constructing c.

We now have the semantics. The syntax is as follows:

...

And finally, the security we are targeting is a notion of Strong Unforgeability under Chosen Message Attack (SUF-CMA). What makes this *strong*? Well, on top of regular UF-CMA, we also require that adversaries cannot come up with a message-signature pair which hasn't even been issued yet.

12 Lecture 13: May 11th

Definition (Decisional LWE Assumption)**.**

13 Lecture 14: May 13th

summary

13.1 Fully-Homomorphic Encryption

We start with introducing a new primitive.

Definition (Secret-key FHE)**.** New primitive that allows us to compute homomorphically.

- Gen outputs (sk, ek), where sk is the secret key and ek is the evaluation key
- $\mathsf{Enc}(sk, \mu)$ = (randomized) encryption of a bit $\mu \in \{0, 1\}$
- $\mathsf{Dec}(sk, c) \in \{0, 1\}$ = decryption of the ciphertext c
- $\mathsf{Eval}(ek, f, c_1, \ldots, c_k)$ where $c_1, \ldots, c_k$ are ciphertexts, and $f : \{0, 1\}^k \to \{0, 1\}$ is a function

Correctness is defined as

$$\mathsf{Dec}(sk, \mathsf{Eval}(ek, f, \mathsf{Enc}(sk, \mu_1), \ldots, \mathsf{Enc}(sk, \mu_k))) = f(u_1, \ldots, u_k)$$

The security property we'd want to define is a bit subtle. First notice that FHEs can *never* be IND-CCA, since we can always create a new ciphertext using the Eval of a function f.

But before that, why do we even care about FHE? Well, for one, it give us a secure outsourcing of computation. Imagine if we wanted to run something on Claude Mythos, which is hosted on the anthropic servers. But I don't want them to know what I am querying.

With FHE, we could first generate some (sk, ek) pair, then send $(\mathsf{Enc}(sk, q), ek)$ to the server. From there, mythos computes $\mathsf{Eval}(ek, f, ct_q)$, then sends back $ct_{f(q)}$. We then decrypt $\mathsf{Dec}(sk, ct_{f(q)})$ which outputs $f(q)$, as desired.

Today, we're going to build FHE from the D-LWE Assumption. Let's take a look at the **GSW Encryption**:

- Gen: Scret key is a vector $s = \begin{bmatrix} -s' \\ 1 \end{bmatrix} \in \mathbb{Z}_q^n$ where $s' \xleftarrow{\$} \mathbb{Z}_q^{n-1}$.
- $\mathsf{Enc}(s, \mu) = A + \mu G$ where $A = \begin{bmatrix} A' \\ (s')^T A' + (e')^T \end{bmatrix}$, $A' \xleftarrow{\$} \mathbb{Z}_q^{(n-1)\times k}$ and $e' \xleftarrow{\$} \chi^k$.

14 Lecture 15: May 18th

summary

14.1 Full-Domain Hash Construction

Recall that when we introduced a FDH, we didn't quite have a "good" construction nor a proof. Today we won't get to a proof, but we will talk about a canonical contruction using our new friend – lattices.

Recall that FDH fundamentally rely on trapdoor permutation, which is essentially a bijective permutation between the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$.

Today, we relax that definition a bit to introduce a **trapdoor function**. Specifically, we say that we no longer require a strict *permutation*, but rather a *surjective* functions. That is, we require $\mathsf{Enc}(PK, M)$ to output an eement of the ciphertext space $\mathcal{C}$, but introduce a new $\mathsf{Inv}(SK, C)$ that outputs some M' such that $\mathsf{Enc}(PK, M') = C$. In other words, many plaintexts can map to the same ciphertext.

In HW4, we'll prove that in the ROM, dense Trapdoor Functions (TDF) are still secure. So today, we'll instatiate a dense TDF from lattices.

Before we get to that, we introduce *another* lattice assumption.

Definition (Short Integer Solution (SIS) Assumption)**.** For every adversary $\mathcal{A}$, define

$$\mathsf{Adv}^{\text{sis}}_{n,k,\beta}(\mathcal{A}) = \Pr\Big[A \xleftarrow{\$} \mathbb{Z}_q^{k\times n} : x \in \mathbb{Z}_q^n \leftarrow \mathcal{A}(A), Ax = 0 \wedge \|x\|_\infty \leq \beta\Big]$$

The SIS_n, k, β assumption: For all PPT $|A$, $\mathsf{Adv}^{\text{sis}}_{n,k,\beta}(\mathcal{A})$ is negligible in n.

Definition (Inhomogeneous Short Integer Solution (I-SIS) Assumption)**.** For every adversary $\mathcal{A}$, define

$$\mathsf{Adv}^{\text{sis}}_{n,k,\beta}(\mathcal{A}) = \Pr\Big[A \xleftarrow{\$} \mathbb{Z}_q^{k\times n}, y \leftarrow \mathbb{Z}_q^k : x \in \mathbb{Z}_q^n \leftarrow \mathcal{A}(A, y), Ax = y \wedge \|x\|_\infty \leq \beta\Big]$$

The $\text{I-SIS}_n, k, \beta$ assumption: For all PPT $|A$, $\mathsf{Adv}^{\text{i-sis}}_{n,k,\beta}(\mathcal{A})$ is negligible in n.

But unfortunately, we are not so confident about the hardness of the I-SIS problem yet, since we do not know how to reduce I-SIS to worst-case lattice problems like Gap-SVP.

Okay, so now how to we build a trapdoor?

Let $B \xleftarrow{\$} \mathbb{Z}_q^{n\times\ell}$ where $\ell = n\lceil\log q\rceil + 2n$ and let $R \xleftarrow{\$} \{0,1\}^{\ell\times m}$ where $m = n\lceil\log q\rceil$. Then,

- The matrix $A = [B \mid BR + G] \in \mathbb{Z}_q^{n\times(\ell+m)}$ is close to uniform by LHL
- .

...

Taking a detour, we define a new relevant latice. For a matrix $A \in \mathbb{Z}_q^{n \times m}$, the set

$$\Lambda^{\perp}(A) = \{z \in \mathbb{Z}^m : Az = 0 \pmod q\}$$

is a lattice:

- $0 \in \Lambda^{\perp}(A)$
- $z_1, z_2 \in \Lambda^{\perp}(A), z_1 + z_2 \in \Lambda^{\perp}(A)$