Course Notes

Gidon Rosalki

Contents

Part I

Lecture 1 — 2025-10-22

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

1 Course overview

This is the second year of this format, before this it was the same name, but different format.

1.1 What is cryptography?

Cryptography is an ancient art, that for many years focused mainly on secret communication. For as long as humans have been communicating, we have wanted to be able to communicate in ways that hides the contents from people who are not meant to know it. The main consumers were military, and intelligence. It generally relied on creativity, and personal skill. From 500BC until the 20th century, there was a complete cycle of design → break → repair → break → repair. We will focus on how one **breaks** this cycle. We will focus on modern cryptography, which underwent a radical change in the 20th century, where it became a science, and covers much more than secret communication. It is now consumed by everyone, and relies on rigorous models, definitions, and proofs.

So, to answer our question: The scientific study of techniques for designing systems that withstand adversarial behaviour.

1.2 Course objectives

We want to introduce the basic paradigms, principles of cryptography, and explore a variety of cryptographic tools and systems. We will learn how to reason about their security, and how to use them correctly. By the end of this course we will be educated crypto consumers, and know why it is dangerous to assume you are a "crypto expert" (spoiler, you're really really not. Do not **ever** roll your own crypto), and be able to learn more about cryptography on our own.

Tentative structure:

1. Weeks 1 - 5: Private key cryptography
2. Weeks 6 - 10: Public key cryptography
3. Weeks 11 - 13: Zero knowledge proofs and secret computation

We are recommended to read

- J. Katz, and Y. Lindell's *Introduction to Modern Cryptography*
- O. Goldreich *Foundations of Cryptography - Volume 1: Basic tools*
- O. Goldreich *Foundations of Cryptography - Volume 2: Basic applications*
- Coursera's *Cryptography* course by Professor Jonathan Katz

There will be somewhere between 3 and 5 homeworks, depending on how bothered the lecturer can be, and our final grade will be made up of 10% the $n-1$ best homeworks, and 90% final exam.

2 Symmetric key encryption

Let there be Alice, and Bob, located in different places, that want to communicate secretly. Eve will observe their communications. Our assumption is that Alice, and Bob, share a secret key, that is not known to Eve. This key is used for both encryption, and decryption. This key is some collection of bits, which may be used as described above. Let us formalise these concepts: An encryption system includes three algorithms: *KeyGen*, *Enc*, *Dec*. Let there be the key space $\mathcal{K}$, plaintext / message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$.

- The key generation algorithm KeyGen outputs a key $k \in \mathcal{K}$
- The encryption algorithm Enc takes a key $k \in \mathcal{K}$, and a plaintext $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{C}$
- Decryption algorithm Dec takes a key $k \in \mathcal{K}$, and a ciphertext $c \in \mathcal{C}$, and outputs a plaintext $m \in \mathcal{M}$

$$\begin{aligned} k &\leftarrow KeyGen() \\ c &\leftarrow Enc_k(m) \\ m &= Dec_k(c) \end{aligned}$$

In this course ← indicates randomised generation, and = indicates deterministic generation.

2.1 Correctness

An encryption system is defined as correct if

$$\forall k \in \mathcal{K}, m \in \mathcal{M} \; Dec_k\left(Enc_k\left(m\right)\right) = m$$

Kerckhoff's principle: All of KeyGen, Enc, and Dec are publicly known, and the only secret is the key k. A crypto system for whom the only security is the secrecy of the algorithms is not secure.

2.2 Caesar Cipher

Let there be:

- KeyGen uniformly samples $k \leftarrow \{0, \ldots, 25\}$
- $M = \{a, \ldots, z\}^l$ and $C = \{A, \ldots, Z\}^l$
- Enc shifts each letter k positions forward (wrapping around from z to a)
- Dec performs the same wrapping shift, but backwards

This is **not** a secure cipher (shocking, I know). Why? There are only 26 possible keys. An important part of good ciphers is that $|K|$ must **not** allow an exhaustive search.

2.3 Substitution cipher

Let there be:

- KeyGen uniformly samples a permutation k over $\{a, \ldots, z\}$
- $M = \{a, \ldots, z\}^l$ and $C = \{A, \ldots, Z\}^l$
- Enc applies the permutation k to each letter
- Dec applies the inverse permutation k^{-1}

This is not secure either (shocking, I know). Despite there being many more keys (26!), this is particularly susceptible to frequency analysis, where we use statistical patterns of the frequencies of different letters in the source language.

2.4 Vigenere cipher

Let there be:

- KeyGen uniformly samples $k = k_0 \ldots k_{t-1} \leftarrow \{0, \ldots, 25\}^t$
- $M = \{a, \ldots, z\}^l$ and $C = \{A, \ldots, Z\}^l$
- Enc shifts the ith letter $k_{i \bmod t}$ positions forward
- Dec applies the inverse shift

Not secure, it is trickier, but since the key is repeated, we can figure out the length of the key, and establish the different parts of the key through frequency analysis once more.

3 Historical ciphers

There is a fascinating history of interesting, creative (and now broken) ideas. It was particularly influenced by world history (e.g. the cryptanalysis of the German enigma in World War 2). Creating a crypto system is very very hard. Trying to do so will probably result in one that is easily broken.

4 Basic principles of modern cryptography

Analysing the security pf a cryptographic system involves:

1. Formalising a precise definition of security (security = computational ability × type of attack × notion of "break")
2. Stating the underlying assumptions: Others will attempt to validate (or invalidate) your assumptions
3. Proving that the definition is satisfied given the assumptions. Despite this, schemes can still be broken.

There are a few attacks on encryption schemes:

- Known ciphertext attack: Eve may observe a challenge ciphertext c^*
- Known plaintext attack: Eve learns pairs $(m, Enc_k(m))$, and then observes a challenge ciphertext c^*
- Chosen plaintext attack (CPA): Eve learns airs $(m, Enc_k(m))$ for messages m of her choice, then observes a challenge ciphertext c^*
- Chosen ciphertext attack: (CCA) Eve learns pairs $(m, Enc_k(m))$, for messages m of her choice, and pairs $(c, Dec_k(c))$ for ciphertexts c of her choice, and then observes a challenge ciphertext $c^* \neq c$

So, what does it mean to break an encryption scheme? Does it mean recovering the key? Recovering the plaintext? Recovering part of the plaintext? Not really any of these. Breaking an encryption scheme means learning anything "meaningful" about the plaintext? So, how do we define "meaningful"? We'll come back to that.
We shall characterise an adversary's computational abilities as follows:

- Typically (not always) run in probabilistic polynomial time (PPT)
- Sometimes, we will say computationally unbounded

5 Perfect secrecy

Let (K:eyGen, Eng, Dec) be a symmetric key encryption scheme. Alice and Bob share a key $k \leftarrow KeyGen()$. Eve knows an a-priori distribution M. Informally, perfect secrecy is that the ciphertext c does not reveal *any* information about the plaintext m.

Definition 5.1 (Perfect secrecy)**.** *A symmetric key encryption scheme $\Pi = (KeyGen, Enc, Dec)$ is* ***perfectly secret*** *if for every distribution over M, and for every $m \in \mathcal{M}$, and for every $c \in \mathcal{C}$ it holds that*

$$Pr\left[M = m | C = c\right] = Pr\left[M = m\right]$$

That is, the probability that some plaintext is the plaintext given the ciphertext, is the same as the probability that some plaintext is the plaintext, with no priors.

Consider a die. If I throw a die, the probability of guessing its result is $\frac{1}{6}$. The encryption system is perfect, if given an encrypted form of what number was thrown, the probability of knowing what number was thrown is still $\frac{1}{6}$.

Lemma 1. *A symmetric key encryption scheme Π is perfectly secret* ***if and only if*** *for every distribution over M, for every $m \in \mathcal{M}$, and for every $c \in \mathcal{C}$, it holds that*

$$Pr\left[C = c | M = m\right] = Pr\left[C = c\right]$$

I.e. the probability of a specific message encoding to a specific ciphertext is the same for every message in the world.

. Let there be a distribution over $M, m \in \mathcal{M}$, and $c \in \mathcal{C}$. Let us assume that

$$Pr\left[C = c | M = m\right] = Pr\left[C = c\right]$$

therefore

$$\begin{aligned} Pr\left[M = m | C = c\right] &= \frac{Pr\left[C = c | M = m\right] \cdot P\left[M = m\right]}{Pr\left[C = c\right]} \\ &= Pr\left[M = m\right] \end{aligned}$$

The other direction is the exact same thing, uses bayes theorem, but swapping M and C. □

Lemma 2. *A symmetric key encryption scheme π is perfectly secret* ***if and only if*** *for every distribution over $\Updownarrow$, for every $m_0, m_1 \in \Updownarrow$ and for every $c \in \rfloor$ it holds that*

$$pr\left[c = c | m = m_0\right] = pr\left[c = c | m = m_1\right]$$

. □

Theorem 1. *The shift and substitution ciphers are* ***not perfectly secret*** *for plaintexts of length $l > 1$.*

. Shift cipher:

$$Pr\left[C = "AB" | M = "ab"\right] = \frac{1}{26} \neq 0 = Pr\left[C = "AB" | M = "aa"\right]$$

□

6 One time pad

Created by Turing.

- $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^l$
- KeyGen uniformly samples $k \leftarrow \{0,1\}^l$
- $Enc_k(m) = m \oplus k$
- $Dec_k(c) = c \oplus k$

This is correct since:

$$\forall k \in \mathcal{K},\ m \in \mathcal{M}\ Dec_k(Enc_k[m]) = Dec_k[m \oplus k] = m \oplus k \oplus k = m$$

Theorem 2 (Perfect secrecy)**.** *The one time pad is perfectly secret for plaintexts of any length l*

. Let us fix $m_0, m_1 \in \mathcal{M}$, and $c \in \mathcal{C}$. We will prove that

$$Pr[C = c | M = m_0] = Pr[C = c | M = m_1]$$

For each $b \in \{0,1\}$ it holds that

$$\begin{aligned} Pr[C = c | M = m_b] &= Pr[M \oplus K = c | M = m_b] \\ &= Pr[m_b \oplus K = c] \\ &= Pr[K = c \oplus m_b] \\ &= \frac{1}{2^l} \end{aligned}$$

This is true for every m_0, m_1, and so is generalised, as is required. □

6.1 Limitations of the one time pad

Keys have to be as long as the plaintexts, and so are very long. Additionally, there is "Two time" insecurity. Given $c = Enc_k(m)$ and $c' = Enc_k(m')$, we can learn $c \oplus c' = m \oplus m'$. There is an additional insecurity against known plaintext attacks. From m and $c = Enc_k(m)$ we can recover $k = m \oplus c$.

Theorem 3. *Let Π be a symmetric encryption scheme, with key space $\mathcal{K}$, and message space $\mathcal{M}$. If Π is perfectly secret, then $|\mathcal{K}| \geq |\mathcal{M}|$*

. Let us assume that $|\mathcal{K}| < |\mathcal{M}|$, and then we will show that the scheme is not perfectly secret. Let M be the uniform distribution over $\mathcal{M}$, and fix some $m \in \mathcal{M}$. Let us also fix some $c \in \mathcal{C}$, which is a possible encryption of m. Let

$$\mathcal{M}(c) \overset{def}{=} \left\{\hat{m} : \hat{m} = Dec_{\hat{k}}(c) \text{ for some } \hat{k} \in \mathcal{K}\right\}$$

Then $|\mathcal{M}(c)| \leq |\mathcal{K}|$. Thus, the assumption that $|\mathcal{K}| < |\mathcal{M}|$ implies that $|\mathcal{M}(c)| < |\mathcal{M}|$. In particular, there exists some $m^* \in \mathcal{M} : m^* \notin \mathcal{M}(c)$. This implies that

$$Pr[M = m^* : C = c] = 0 \neq \frac{1}{|\mathcal{M}|} = Pr[M = m^*]$$

and so the scheme is not perfectly secret. □

6.2 Characterising perfect secrecy

Theorem 4 (Shannon's theorem)**.** *Let Π be a symmetric key encryption scheme for which $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$. Π is perfectly secret* ***if and only if*** *the following two conditions hold:*

1. *Every $k \in \mathcal{K}$ chosen by KeyGen is chosen with a probability of $\frac{1}{|\mathcal{K}|}$*
2. *For every $m \in \mathcal{M}$, and $c \in \mathcal{C}$, there exists exactly one $k \in \mathcal{K}$ such that $Enc_k(m)$ outputs c*

7 Tutorial

Behold: Another definition of perfect secrecy:

Exercise 1 (Perfect secrecy)**.** *For every encryption system* Π*, that has perfect secrecy: For every distribution* M *on the plaintext space* $\mathcal{M}$*, and for every 2 ciphertexts* $c_0, c_1 \in \mathcal{C}$*, it is true that*

$$Pr\left[C = c_0\right] = Pr\left[C = c_1\right]$$

Solution. This is incorrect. Let

$$KeyGen : k \leftarrow \{0,1\}^l$$

Let there be an additional bin $b = \begin{cases} 0, & \text{with probability } \frac{1}{3} \\ 1, & \text{with probability } \frac{2}{3} \end{cases}$ $Enc\left(k, m\right) : c = k \oplus m \| b$ here the double line indicates appending

$$Dec\left(c, k\right) : c' \oplus k = m \text{ where } c' = c \text{ without } b$$

So, as we can see here, it does not hold that $Pr\left[C = c_0\right] = Pr\left[C = c_1\right]$, since $\frac{2}{3}$rds of the ciphertexts will end with 1, and $\frac{1}{3}$ will end with 0. □

Exercise 2. *Given* Π *that has perfect secrecy, distribution* M *on* $\mathcal{M}$*. Let there be 2 messages* $m_0, m_1 \in \mathcal{M}$*. For every* $c \in \mathcal{C}$*:*

$$Pr\left[C = c | M = m_0\right] = Pr\left[C = c | M = m_1\right]$$

Solution. Correct: By using Bayes law:

$$Pr\left[C = c | M = m_0\right] = Pr\left[C = c\right]$$
$$Pr\left[C = c | M = m_1\right] = Pr\left[C = c\right]$$

INCOMPLETE As required □

Exercise 3. *Let us define* $\hat{OTP}$*, which is the same as OTP, but the keygen is defined as follows*

$$KeyGen : k \leftarrow \{0,1\}^l \setminus \left\{0^l\right\}$$

Is this secure?

Solution. No. Let there be $M \leftarrow \{0,1\}^l$, and so

$$Pr\left[M = m | C = m\right] = 0 \neq \frac{1}{|M|} = Pr\left[M = m\right]$$

when $M \leftarrow \{0,1\}^l$ □

Part II

Lecture 2 - Private key encryption — 2025-10-29

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

8 Reminder

Last week we discussed symmetric encryption, and perfect secrecy:

$$\mathbb{P}\left[M = m | C = c\right] = \mathbb{P}\left[M = m\right]$$

which has the limitations of only considering security for a single message, and that the key must be as long as the message.

9 Computational security

Computational security is that all the information is present, given $Enc_k\left(m\right)$ one may completely determine k and m. It should be **computationally infeasible** to retrieve any useful information. Here we have two realistic relaxations compared to last week:

1. Security is preserved only against **computationally bounded** adversaries (e.g. 2000 years using currently technology)
2. Allow such adversaries to succeed with some *negligible* probability (small enough that it will essentially never happen)

9.1 Approaches

9.1.1 Concrete approach

Definition 9.1. *A scheme is (t, ε)-secure if every adversary, running for time at most t succeeds in breaking the scheme with probability at most ε.*

We have some sample parameters of $t = 2^{60}$, which is the order of the number of seconds since the big bang, and $\varepsilon = 2^{-60}$, which is order of occurring once every 100 billion years.
This is very useful in practice, and may be tailored to specific technology. However, in general we would like a notion of security that is essentially independent of the underlying technology.

9.1.2 Asymptotic approach

Definition 9.2. *A scheme is secure if every **probabilistic polynomial-time** (PPT) adversary succeeds in breaking the scheme with only **negligible** probability.*

Definition 9.3 (PPT)**.** *An algorithm A runs in **probabilistic polynomial-time** if there exists a polynomial $p\left(\cdot\right)$ such that, for any input $x \in \{0,1\}^*$, and a random tape $r \in \{0,1\}^*$, the computation of $A\left(x; r\right)$ terminates within $p\left(|x|\right)$ steps*

The security parameter:

- KeyGen takes as input the security parameter 1^n, and outputs $k \in \mathcal{K}_n$
- Keys produced by $KeyGen\left(1^n\right)$ should provide security against adversaries whose running time is polynomial in n (so increasing n provides better security)
- $\mathcal{K} = \bigcup_{n \in \mathbb{N}} \mathcal{K}_n,\ \mathcal{M} = \bigcup_{n \in \mathbb{N}} \mathcal{M}_n,\ \mathcal{C} = \bigcup_{n \in \mathbb{N}} \mathcal{C}_n$

Definition 9.4 (Negligible)**.** *A function $f : \mathbb{N} \to \mathbb{R}^+$ is **negligible** if for every polynomial $p\left(\cdot\right)$ there exists an N such that $\forall n > N$, it holds that $f\left(n\right) < \dfrac{1}{p\left(n\right)}$*

For example, $2^{-n}, 2^{-\sqrt{n}}, 2^{-\log^2(n)}$ are all negligible functions, where $\dfrac{1}{2}, \dfrac{1}{\log^2\left(n\right)}, \dfrac{1}{n^5}$ are non negligible.

Theorem 5. *Let $v_1\left(n\right), v_2\left(n\right)$ be negligible functions. Then, for any positive polynomial $p\left(n\right)$, the function $p\left(n\right) \cdot \left(v_1\left(n\right) + v_2\left(n\right)\right)$ is negligible.*

Proof . A negligible function is $\frac{1}{\hat{p}}$, where $\hat{p}$ is larger than every polynomial. As a result, whatever we put in the numerator, will not impact our result. Therefore, the sum remains a negligible function. Multiplying by a polynomial is like writing $\frac{1}{\frac{\hat{p}}{p}}$, or subtracting in the powers: n^{l-c}, but since l is asymptotically larger than all polynomials, we still have a negligible function. □

So why these choices? "Efficient": PPT, and "negligible": smaller than any inverse polynomial. It is intuitively well-behaved under composition:

$$poly\left(n\right)\cdot poly\left(n\right)=poly\left(n\right)$$

Polynomially many invocations of a PPT algorithm is still a PPT algorithm.

$$poly\left(n\right)\cdot negligible\left(n\right)=negligible\left(n\right)$$

Polynomially many invocations of a PPT algorithm that succeeds with a negligible probability is an algorithm that succeeds with a negligible probability overall.

10 Indistinguishable encryptions

The most basic notion of security for symmetric-key encryption: Encryptions of any two messages should be indistinguishable. The adversary still observes only a single ciphertext.

$$Enc_k\left(m_0\right)\approx Enc_k\left(m_1\right)$$

This seems weaker compared to perfect secrecy. Perfectly-secure encryption reveals no information, so intuitively, what security does indistinguishable encryptions provide?

Given $\Pi=\left(KeyGen, Enc, dec\right)$, and an adversary $\mathcal{A}$, consider the experiment $IND_{\Pi,\mathcal{A}}\left(n\right)$, where one of two plaintexts m_0, m_1 is encrypted by the system, and then $\mathcal{A}$ needs to figure out which plaintext it was from the returned ciphertext c. The system is indistinguishable if $\mathcal{A}$ cannot do better than a coin flip.

Definition 10.1 (Indistinguishable encryption). Π *has indistinguishable encryption if for every PPT adversary* $\mathcal{A}$ *there eixsts a negligible function* $v\left(\cdot\right)$ *such that*

$$\mathbb{P}\left[IND_{\Pi,\mathcal{A}}\left(n\right)=1\right]\leq\frac{1}{2}+v\left(n\right)$$

where the probability is taken over the random coins used by $\mathcal{A}$*, and by the experiment.*

Recall the one time pad:

- $\mathcal{K}=\mathcal{M}=\mathcal{C}=\left\{0,1\right\}^l$
- KeyGen uniformly samples $k\leftarrow\left\{0,1\right\}^l$
- $Enc_k\left(m\right)=m\oplus k$, and $Dec_k\left(c\right)=c\oplus k$

Perfectly secure since $\mathbb{P}\left[M=m|C=c\right]=\mathbb{P}\left[M=m\right]$, but requires the key k to be as long as the message m. Way too long. Can we guarantee computational security with shorter keys?

11 Pseudo-random generator

Our goal is to expand a short, random seed into a long "random looking" value:

$$G:\left\{0,1\right\}^l\to\left\{0,1\right\}^l$$

"Random looking" means "indistinguishable" from the uniform distribution.

Definition 11.1 (PRG). *Let* $G:\left\{0,1\right\}^l\to\left\{0,1\right\}^l$ *be a polynomial-time computable function, and let* $l\left(\cdot\right)$ *be a polynomial such that for any input* $s\in\left\{0,1\right\}^n$*, we have* $G\left(s\right)\in\left\{0,1\right\}^{l\left(n\right)}$*. Then,* G *is a* ***pseudorandom generator*** *if the following two conditions hold:*

- *Expansion:* $l\left(n\right)>n$
- *Pseudorandomness: For every PPT "distinguisher"* $\mathcal{D}$*, there exists a negligible function* $v\left(\cdot\right)$ *such that*

$$\left|\mathbb{P}_{s\leftarrow\left\{0,1\right\}^n}\left[\mathcal{D}\left(G\left(s\right)\right)=1\right]-\mathbb{P}_{r\leftarrow\left\{0,1\right\}^{l\left(n\right)}}\left[\mathcal{D}\left(r\right)=1\right]\right|\leq v\left(n\right)$$

The notation $x\leftarrow\left\{0,1\right\}^m$ denotes that x is sampled from the **uniform distribution** over $\left\{0,1\right\}^m$ (so each value is obtained with the probability $\frac{1}{2^m}$)

11.1 Do PRGs even exist?

If so, then how difficult is it to construct a PRG? Recall two properties:

- Expansion: $|G(s)| > |s|$
- Pseudorandomness: For every PPT $\mathcal{D}$, there exists a negligible $v(\cdot)$ such that:

$$\left|\mathbb{P}_{s\leftarrow\{0,1\}^n}\left[\mathcal{D}(G(s)) = 1\right] - \mathbb{P}_{r\leftarrow\{0,1\}^{l(n)}}\left[\mathcal{D}(r) = 1\right]\right| \leq v(n)$$

Let us try. Consider the following candidates that expand a seed $s = s_1 \dots s_n \in \{0,1\}^n$ by a single bit. Let us define

$$G(s) = s0$$

Is it distinguishable from a truly random string? Yes. A truly random string may finish in 1, whereas this may not.

How about

$$G(s) = s_1 \dots s_n s_1$$

It is distinguishable, since we can just check if we begin with the same bit as with which we started.

Finally:

$$G(s) = s_1 \dots s_n z \ : \ z = s_1 \oplus \dots \oplus s_n$$

This is also distinguishable, since we can just check if the final bit is the xor of the bits before it.

The existence of any PRG implies $P \neq NP$. Constructions are known based on various computational assumptions. We have not created a PRG that may be proven to be such, since that would then prove that $P \neq NP$.

Theorem 6. *Let there be $G : \{0,1\}^n \to \{0,1\}^{2n}$. There exists $\mathcal{D}$ (not computationally efficient) such that*

$$\mathbb{P}_{s\leftarrow\{0,1\}^n}\left[\mathcal{D}(G(s) = 1)\right] - \mathbb{P}_{r\leftarrow\{0,1\}^{2n}}\left[\mathcal{D}(r) = 1\right] > \frac{1}{2}$$

Proof . Where $|z| = 2n$,

$$D(z) = Im(G) = \left\{l \in \{0,1\}^{2n} | \exists s \mid G(s) = l\right\}$$

We are asking if it is true that $z \in Im(G)$. If so, we will return 1, and otherwise 0. This distinguisher works since

$$\mathbb{P}[D(G(s)) = 1] = 1$$
$$\mathbb{P}[D(r) = 1] = \frac{|Im(G)|}{2^{2n}} \leq \frac{1}{2^n}$$

□

Useful fact: All efficiently-testable statistical properties of the uniform distribution are preserved by the output of any PRG. For example: If G is a PRG, then there exists a negligible function $v(\cdot)$ such that

$$\mathbb{P}_{s\leftarrow\{0,1\}^n}\left[\text{Fraction of 1s in } G\left(s < \frac{1}{4}\right)\right] \leq v(n)$$

12 PRG-based OTP

Let us assume that there exists PRGs. Let G be a PRG with expansion $l(n)$. $\mathcal{K}_n = \{0,1\}^n$, but $\mathcal{M}_n = \mathcal{C}_n = \{0,1\}^{l(n)}$. $KeyGen(1^n)$ samples $k \leftarrow \{0,1\}^n$. $Enc_k(m) = m \oplus G(k)$, and $Dec_k(c) = c \oplus G(k)$.

So, given k, we generate $G(k)$, where $|G(k)| > |k|$, and then $c = G(k) \oplus m$.

Theorem 7. *If G is a PRG, then the scheme has indistinguishable encryptions.*

Proof . It's not perfect, the key is smaller than the messages, and we proved that the key must be the same length as the messages. We shall prove by **reduction**:

- Given an adversary $\mathcal{A}$, for the encryption scheme, construct a distinguisher $\mathcal{D}$ for the PRG
- $\mathcal{D}$ internally emulates $\mathcal{A}$
- $\mathcal{D}$'s efficiency, and advantage are Polynomially related to $\mathcal{A}$'s

So in short, if G is a PRG, then Π has indistinguishable encryptions. We will prove by contradiction, by assuming that Π is not IE, and therefore G is not a PRG (but we know this to be false, and thus have a contradiction).

Behold, the actual proof: Let us assume that there exists a PPT adversary $\mathcal{A}$ and a polynomial $pp(n)$ such that

$$\mathbb{P}\left[IND_{\Pi,\mathcal{A}}(n)=1\right] \geq \frac{1}{2}+\frac{1}{p(n)}$$

for infinitely many ns.

We will show that there exists a PPT distinguisher $\mathcal{D}$ and a polynomial $q(n)$, such that

$$\left|\mathbb{P}_{s\leftarrow\{0,1\}^n}\left[\mathcal{D}(G(s))=1\right]-\mathbb{P}_{r\leftarrow\{0,1\}^{l(n)}}\left[\mathcal{D}(r)=1\right]\right| \geq \frac{1}{q(n)}$$

for infinitely many ns. Or in short, if there exists the adversary, then we can use it to construct the distinguisher. The distinguisher $\mathcal{D}$, on input z invokes $\mathcal{A}$, and obtains (m_0,m_1). It samples $b\leftarrow\{0,1\}$, and let $b'=\mathcal{A}(z\oplus m_b)$. It then outputs 1 **if and only if** $b'=b$.

From here we get 2 cases:

- Case 1: $z\leftarrow\{0,1\}^{l(n)}$. $\mathcal{A}$'s view is independent of b, and so

$$\mathbb{P}_{z\leftarrow\{0,1\}^{l(n)}}\left[\mathcal{D}(z)=1\right]=\frac{1}{2}$$

- Case 2: $z=G(k)$, where $k\leftarrow\{0,1\}^n$. $\mathcal{A}$'s view is identical to the experiment $IND_{\Pi,\mathcal{A}}$ and so it is equivalent to trying to find if they are distinguishable:

$$\mathbb{P}_{k\leftarrow\{0,1\}^n}\left[\mathcal{D}(G(k))=1\right]=\mathbb{P}\left[IND_{\Pi,\mathcal{A}}(n)=1\right] \geq \frac{1}{2}+\frac{1}{p(n)}$$

So overall we constructed a PPT distinguisher $\mathcal{D}$ such that

$$\left|\mathbb{P}_{s\leftarrow\{0,1\}^n}\left[\mathcal{D}(G(s))=1\right]-\mathbb{P}_{r\leftarrow\{0,1\}^{l(n)}}\left[\mathcal{D}(r)=1\right]\right| \geq \frac{1}{p(n)}$$

which contradicts the theorem that G is a PRG. □

We have made significant progress, but we still have the problem that each key may only be used once.

13 Indistinguishable encryptions revisited

So far, this has enabled it to be infeasible to distinguish between $Enc_k(m_0)$ and $Enc_k(m_1)$, but can we learn information of m from $Enc_k(m)$

Theorem 8 (Toy theorem)**.** *Let Π have indistinguishable encryptions. Then, for any PPT adversary $\mathcal{B}$, there exists a negligible function $v(\cdot)$ such that*

$$\mathbb{P}\left[\mathcal{B}(1^n,Enc_k(m))=LSB(m)\right] \leq \frac{1}{2}+v(n)$$

where $m\leftarrow\{0,1\}^{l(n)}$ is sampled uniformly

Proof by reduction. Assume a contradiction that there exists a PPT adversary $\mathcal{B}$ and a polynomial $p(n)$ such that

$$\mathbb{P}\left[\mathcal{B}(1^n,Enc_k(m))=LSB(m)\right] \leq \frac{1}{2}+\frac{1}{p(n)}$$

for infinitely many ns. We then show that there exists a PPT adversary $\mathcal{A}$ and a polynomial $q(n)$ such that

$$\mathbb{P}\left[IND_{\Pi,\mathcal{A}}(n)=1\right] > \frac{1}{2}+\frac{1}{q(n)}$$

for infinitely many ns.

Behold the proof: For each $\sigma\in\{0,1\}$, let $I_\sigma\subset\{0,1\}^l$ be the set of messages whose LSB is σ. We will create the adversary $\mathcal{A}$, which on input 1^n will sample $m_0\leftarrow I_0$, and $m_1\leftarrow I_1$ uniformly and independently. On input c^*, it will output $b'=B(1^n,c^*)$. In short, $\mathcal{A}$ creates 2 messages, receives the encrypted form of one of them, and gives it to $\mathcal{B}$. If $\mathcal{B}$ correctly assumes the LSB, then we win, if not, we fail.

$$\begin{aligned}\mathbb{P}\left[IND_{\Pi,\mathcal{A}}(n)=1\right]&=\mathbb{P}\left[\mathcal{B}(1^n,Enc_k(m_b))=b\right]\\&=\frac{1}{2}\mathbb{P}_{m_0\leftarrow I_0}\left[\mathcal{B}(1^n,Enc_k(m_0))=0\right]+\frac{1}{2}\mathbb{P}_{m_1\leftarrow I_1}\left[\mathcal{B}(1^n,Enc_k(m_1))=1\right]\\&=\mathbb{P}_{m\leftarrow\{0,1\}^l}\left[\mathcal{B}(1^n,Enc_k(m))=LSB(m)\right] \geq \frac{1}{2}+\frac{1}{p(n)}\end{aligned}$$

□

13.1 Semantic security

Goldwasser-Micali in 1982: "Whatever" can be computed efficiently, given the ciphertext, can essentially be computed efficiently without the ciphertext.

Definition 13.1 (Semantically secure)**.** Π *is **semantically secure** if for every adversary $\mathcal{A}$ there exists a PPT "simulator" $\mathcal{S}$ such that for every efficiently sampleable plaintext distribution $M = \{M_n\}_{n\in\mathbb{N}}$, and all polynomial-time computable functions f and h, there exists a negligible function $v(\cdot)$ such that*

$$\left|\mathbb{P}\left[\mathcal{A}\left(1^n, Enc_k(m), h(m)\right) = f(m)\right] - \mathbb{P}\left[\mathcal{S}\left(1^n, h(m)\right) = f(m)\right]\right| \leq v(n)$$

where $k \leftarrow KeyGen\left(1^n\right)$ and $m \leftarrow M_n$

Or in other words, whatever you can learn from the encryption, can also be efficiently learnt *without* the encryption, or most simply, the ciphertext teaches us **nothing**.

Theorem 9. Π *is **semantically secure if and only if** it has **indistinguishable encryption***

Why do we need both notions? Well, semantic security explains "what security means", where indistinguishability of encryption is "easier with which to work". Since they are equivalent, we can use IE, to show semantic security.

13.2 One way functions

Definition 13.2. *A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is **one way** if for any PPT A*

$$\mathbb{P}_{y\leftarrow f(U_n)}\left[A\left(1^n, y\right) \in f^{-1}(y)\right] \leq negligible\,(y)$$

In short, easy to compute, but hard to invert on a random image.

Informal theorem: One way functions are the basis for foundational cryptography. They are **complete** for private key cryptography (PRG ⇔ OWF, as in, one can make PRGs from one way functions, and vice versa)

Some recommended reading: J. Katz and Y. Lindell. Introduction to Modern Cryptography. Chapter 3 (Private-Key Encryption): 3.0 – 3.3

Part III

Lecture 3 - Private key encryption II — 2025-11-05

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

14 Recap

Last week we discussed computational secrecy, which includes indistinguishable encryptions, semantic security. To achieve this we used the tool of Pseudorandom Generators (PRGs), and investigated One time pads that use PRGs. These have short keys, but each key can still only be used once.

15 Computational Indistinguishability

Two probability distributions $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if no "efficient" algorithm "can tell them apart".

Example I (PRG G):

$$\begin{aligned} X_n &= G(s) \text{ for } s \leftarrow \{0,1\}^n \\ Y_n &= \text{ uniform distribution over } \{0,1\}^{l(n)} \end{aligned}$$

Example II (IND-secure): Let $\Pi = (KeyGen, Enc, Dec)$:

$$X_n = Enc_k(000) \text{ for } k \leftarrow KeyGen(1^n) Y_n \qquad = Enc_k(101) \text{ for } k \leftarrow KeyGen(1^n)$$

Definition 15.1 (Computationally indistinguishable)**.** *Two probability distributions $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are **computationally indistinguishable** if for every PPT distinguisher D there exists a negligible function $v(\cdot)$ such that*

$$\left| \Pr_{x\leftarrow X_n}[D(1^n, x) = 1] - \Pr_{y\leftarrow Y_n}[D(1^n, y) = 1] \right| \leq v(n)$$

This is denoted as $X \approx^c Y$.

Theorem 10. *Let $G : \{0,1\}^n \to \{0,1\}^{4n}$ be a PRG, then $H(s_1, s_2) = G(s_1) || G(s_2)$ is a PRG.*

Proof . Our paradigm for this kind of proof is *reduction* via a *hybrid argument.*
Reduction: Given a distinguisher D, for H, construct a distinguisher A for G.
Hybrid argument: Let us suppose that between $G(s_1), G(s_2)$ D has advantage ε. Let us create a new PRG, that given s_1, s_2, ignores s_2, and returns $G(s_1), r_2$. So, between $G(s_1), G(s_2)$ and $G(s_1), r_2$, it holds that D has at least the advantage $\frac{\varepsilon}{2}$, or between $G(s_1), r_2$ and r_1, r_2 it holds that D has the advantage of at least $\frac{\varepsilon}{2}$. So:

$$\begin{aligned} \varepsilon &\leq |\mathbb{P}[D(G(s_1)||G(s_2)) = 1] - \mathbb{P}[D(r_1||r_2) = 1]| \\ &\leq |\mathbb{P}[D(G(s_1)||G(s_2)) = 1] - \mathbb{P}[D(G(s_1)||r_2) = 1]| + |\mathbb{P}[D(G(s_1)||r_2) = 1] - \mathbb{P}[D(r_1||r_2) = 1]| \end{aligned}$$

Let us define A, which on input $z \in \{0,1\}^{4n}$ with sample $s_1 \leftarrow \{0,1\}^n$ and output $D(G(s_1)||z)$. In this case, we have created an adversary that distinguishes between the first 2 cases based off the difference of $G(s_2)$ and r_2. We may similarly create a second adversary that performs the same, and outputs $D(Z||r_2)$. Since *one* of these transitions must be distinguishable with an advantage of at least $\frac{\varepsilon}{2}$, we have found an adversary A for G, which is a contradiction to the given that G is a PRG. □

16 Security against a CPA

Given $\Pi = (KeyGen, Enc, Dec)$ and an adversary $\mathcal{A}$, we considered the experiment $IND_{\Pi,\mathcal{A}}(n)$. Does this experiment model realistic attacks? Not especially, since it assumes that it can only observe a single encryption, which is not especially realistic.

CPA is an extension of this system, where once again, we choose a key $k \leftarrow KeyGen(1^n)$, $\mathcal{A}$ provides m_0, m_1, we choose $b \leftarrow \{0,1\}$, return $c^* \leftarrow Enc_k(m_b)$, and $\mathcal{A}$ needs to choose b'. Here, $\mathcal{A}$ provides as many ms for encryption as it likes, which are encrypted by k. Then it provides m_0, m_1, b is chosen, and $c^* \leftarrow Enc_k(m_b)$ is returned to $\mathcal{A}$. $\mathcal{A}$ can then request more encryptions of ms, and then has to return b', which value it thought b to be. Note, OTP does *not* stand up to this type of attack, nor does any other deterministic encryption system.
It may also be said that $\mathcal{A}$ has access to an encryption oracle, denoted $\mathcal{A}^{Enc_k(\cdot)}$.

Definition 16.1 (CPA secure)**.** Π *has* ***indistinguishable encryptions under a chosen plaintext attack*** *if for every PPT adversary $\mathcal{A}$ there exists a negligible function $v(\cdot)$ such that*

$$\Pr\left[IND^{CPA}_{\Pi,\mathcal{A}}(n) = 1\right] \leq \frac{1}{2} + v(n)$$

So, a Π that is CPA secure must used a **randomised** encryption algorithm. We will also provide the notation

$$IND^{CPA}_{\Pi,\mathcal{A}}(n) = \begin{cases} 1, & \text{if } b' = b \\ 0, & \text{otherwise} \end{cases}$$

We shall ask, is CPA security "too strong"? Well, no, since adversaries may often know, influence, or even determine the encrypted content, and CPA security captures all such influences.

17 Pseudorandom functions

A pseudorandom function is a function that "looks like" a truly random function. What is a truly random function? Let there be the sets of all the functions in the world (relevant to us):

$$Func_{n\to l} = \text{ the set of all functions from } \{0,1\}^n \text{ to } \{0,1\}^l$$
$$|Func_{n\to l}| = \left|\{0,1\}^l\right|^{|\{0,1\}^n|} = 2^{l\cdot 2^n}$$

A truly random function is a function h, sampled uniformly from $Func_{n\to l}$. For each $x \in \{0,1\}^n$ the value $h(x) \in \{0,1\}^l$ is chosen uniformly and independently of all other x's.

Pseudorandom functions are an efficiently computable keyed function

$$F_k(\cdot) : \{0,1\}^n \to \{0,1\}^l$$

that is *indistinguishable* from a truly random function. Our distinguisher $\mathcal{D}$ needs to provide an input x, and then distinguish whether or not it received in return $h(x)$, with h sampled uniformly from $Func_{n\to l}$, or if it received $F_k(x)$, where k is sampled uniformly from $\{0,1\}^n$.

Definition 17.1 (PRF)**.** *An efficiently computable keyed function*

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

is ***pseudorandom*** *if for every PPT distinguisher $\mathcal{D}$ there exists a negligible function $v(\cdot)$ such that*

$$\left|\Pr\left[\mathcal{D}^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[\mathcal{D}^{h(\cdot)}(1^n) = 1\right]\right| \leq v(n)$$

where $k \leftarrow \{0,1\}^n$ and $h \leftarrow Func_{n\to l}$

This defines the security of PRFs with respect to *uniformly distributed* keys $k \in \{0,1\}^n$. More generally, keys may be of any length, and not uniformly distributed. This is captured via a PPT key-generation algorithm $k \leftarrow KeyGen(1^n)$.

The methodology for using PRFs is as follows:

1. Prove security assuming a truly random function is used
2. Prove that if an adversary can break the scheme when PRF is used, then it can be used to distinguish the PRF from a truly random function

18 CPA secure encryptions from PRFs

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^l$ be a PRF.

- Key generation: Sample $k \leftarrow \{0,1\}^n$
- Encryption: On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^l$, sample $r \leftarrow \{0,1\}^l$ and we get output

$$c = (r, F_k(r) \oplus m)$$

- Decryption: On input $k \in \{0,1\}^n$ and $c = (r, s)$, output $m = F_k(r) \oplus s$

Theorem 11. *If F is a PRF, then the scheme Π_F above is CPA secure*

Proof . We will ebgin by showing that this is a correct encryption scheme:

$$\begin{aligned} r &\leftarrow \{0,1\}^n \\ Enc\left(k,m\right) &= \left(r, F_k\left(r\right) \oplus m\right) \\ Dec\left(k,\left(r,s\right)\right) &= F_k\left(r\right) \oplus s \\ &= F_k\left(r\right) \oplus \left(F_k\left(r\right) \oplus m\right) \\ &= m \end{aligned}$$

So why does encrypting the same message twice result in a different c? Due to r changing every time, and so $F_k\left(r\right)$ is a different, completely random seeming series of bits, every time. Consider $m = 0$, encrypted many different times. The results will be:

$$\begin{gathered} \left(r_1, F_k\left(r_1\right)\right) \\ \left(r_2, F_k\left(r_2\right)\right) \\ \left(r_3, F_k\left(r_3\right)\right) \\ \vdots \end{gathered}$$

Which are all different, completely random seeming series of bits.

So, how will we prove this? Reduction! As always!
Consider the above scheme, called Π_F. Let us define the scheme Π_h, where instead of using F_k, we use h. We will note that Π_h is not efficient, and will not be used in reality, since it uses a truly random function. We will show that Π_h is CPA, and that we cannot distinguish between Π_h and Π_F, and thus Π_F is also CPA.

Let $\mathcal{A}$ be a PPT adversary, then

$$\begin{aligned} \Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] \leq &\left|\Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] - \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right]\right| \\ &+ \mathbb{P}\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right] \end{aligned}$$

We will begin by claiming that there exists a negligible $v\left(n\right)$ such that

$$\left|\Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] - \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right]\right| \leq v\left(n\right)$$

We will additionally claim: Let $q\left(n\right)$ be the number of queries made by $\mathcal{A}$ to the encryption oracle, then

$$\Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right] \leq \frac{1}{2} + \frac{q\left(n\right)}{2^n}$$

So,

$$\begin{aligned} \Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] \leq &\left|\Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] - \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right]\right| \\ &+ \mathbb{P}\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right] \\ &\leq \frac{1}{2} + \left(\frac{q\left(n\right)}{2^n} + v\left(n\right)\right) \end{aligned}$$

Let us begin with the first claim: There exists negligible $v\left(n\right)$

$$\left|\Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] - \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right]\right| \leq v\left(n\right)$$

and prove by reduction: Given an adversary $\mathcal{A}$ for the encryption scheme, construct a distinguisher $\mathcal{D}$ for the PRF. $\mathcal{D}$ has oracle access to a function $\mathcal{O}$, which is either $F_k\left(\cdot\right)$, or a truly random $h\left(\cdot\right)$. $\mathcal{D}$ emulates the CPA experiment to $\mathcal{A}$, and observes whether $\mathcal{A}$ succeeds. If $\mathcal{A}$ succeeds, $\mathcal{D}$ outputs 1, else, $\mathcal{D}$ outputs 0.
Let us assume the contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p\left(n\right)$ such that

$$\left|\Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right] - \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right]\right| \geq \frac{1}{p\left(n\right)}$$

for infinitely many ns. The distinguisher $\mathcal{D}^{\mathcal{O}}$ will sample $b \leftarrow \{0,1\}$, and invoke $\mathcal{A}$. It will respond to $\mathcal{A}$'s encryption, and challenge queries, using $\mathcal{O}$, and output 1 **if and only if** $b' = b$. This leaves us with two cases:

1. $\mathcal{O} = F_k$ where $k \leftarrow \{0,1\}^n$. $\mathcal{A}$'s view is identical to $IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right)$

$$\Pr\left[D^{F_k(\cdot)}\left(1^n\right) = 1\right] = \Pr\left[IND^{CPA}_{\Pi_F,\mathcal{A}}\left(n\right) = 1\right]$$

2. $\mathcal{O} = h$, where $h \leftarrow Func_{n\to l}$. Here, $\mathcal{A}$'s view is identical to $IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right)$ and

$$\Pr\left[D^{h(\cdot)}\left(1^n\right) = 1\right] = \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}\left(n\right) = 1\right]$$

Thus a contradiction, and we have shown claim 1.

Claim 2: Let $q(n)$ be the number of queries made by $\mathcal{A}$ to the encryption oracle, then

$$\Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}(n) = 1\right] \leq \frac{1}{2} + \frac{q(n)}{2^n}$$

Each encryption query m_i is answered with $(r_i, h(r_i) \oplus m_i)$ for a uniform, and independently chosen $r_i \leftarrow \{0,1\}^n$. The *repeat* is the event in which r^* is used at least once by the encryption oracle (ie $r^* = r_i$). If *repeat* does not occur, then $h(r^*)$ is completely uniform and independent of $\mathcal{A}$'s view, and therefore $\mathcal{A}$'s view is independent of b:

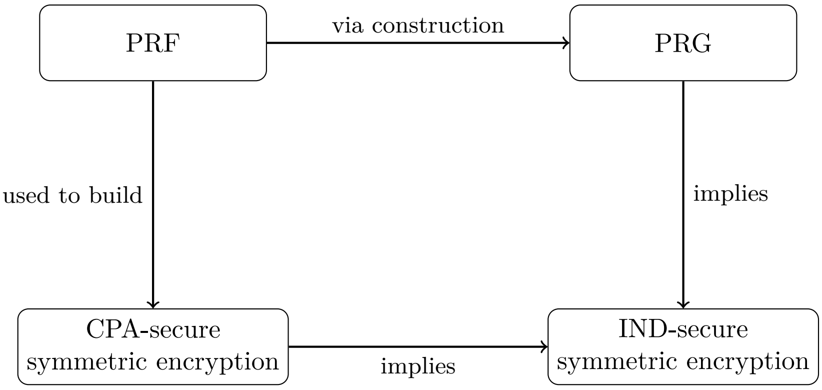
$$\Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}(n) = 1 | \overline{\mathit{Repeat}}\right] = \frac{1}{2}$$

So

$$\begin{aligned}\Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}(n) = 1\right] &\leq \Pr\left[IND^{CPA}_{\Pi_h,\mathcal{A}}(n) = 1 | \overline{\mathit{Repeat}}\right] + \Pr\left[\mathit{Repeat}\right] \\ &= \frac{1}{2} + \frac{q(n)}{2^n}\end{aligned}$$

□

Our world of crypto primitives so far is from PRF, we can build both PRG, and CPA-secure symmetric key encryption, and from both PRG, and CPA-secure symmetric key encryption, we can build IND-secure symmetric key encryption.



19 Practical heuristics block ciphers

In practice, block ciphers are designed to be secure instantiations of pseudorandom permutations (PRPs). A block cipher is an efficiently-computable keyed permutation

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

They have concrete security rather than asymptotic security, and a block cipher is considered "secure" if the best known "attack" requires time roughly 2^n (≈brute-force search for the key).

DES was the The Data Encryption Standard. It was developed in the 1970s by IBM, with help from the NSA, and adopted in 1977. The key length is 56 bits, with block length of 64 bits. The best known attack in practice is essentially a brute force key search, which is eminently possible by your smartphone, since the key length is so short. Thus it is no longer considered secure. It remains widely used in the strengthened form 3DES:

$$3DES_{k_1,k_2,k_3}(x) = DES_{k_1}\left(DES^{-1}_{k_2}\left(DES_{k_3}(x)\right)\right)$$

So there are $3 \cdot 56$ bit keys, but can be broken in $2^{2\cdot 56}$. It is also (unsurprisingly) slower than DES.

These days, we have moved on from DES to AES. In 1997 NIST published a call fro candidate block ciphers to replace DES. 15 candidates were proposed, each being extensibely analysed by the public and other teams. The winner (originall called Rijndael) was announced in late 2000, and was chosen based on security, efficiency, ability to implement in hardware, and so on. It uses key lengths of 128/192/256, with a block length of 128 bits. To date, there are no known practical attacks better than brute-force key search. It is massively widely used, with all modern hardware having built in optimisations. Recall the definition of CPA secure encryption from any PRF:

$$Enc_k(m; r) = (r, F_k(r) \oplus m)$$

In practice, AES as a PRF enables one to encrypt 128 bit blocks:

$$Enc_k(m; r) = (r, AES_k(r) \oplus m)$$

Why not simply $AES_k(m)$? AES is deterministic, and so susceptible to CPA attacks.

In order to encrypt long messages, we break them up into 128 bit blocks:

$$Enc_k\left(m_1 \dots m_l; r_1 \dots r_l\right) = \left(r_1, AES_k\left(r_1\right) \oplus m_1\right) \dots \left(r_l, AES_k\left(r_l\right) \oplus m_l\right)$$

This has the drawback of the ciphertext length being twice the plaintext. This can be improved by changing the structure slightly:

$$Enc_k\left(m_1 \dots m_l; r_1\right) = \left(r, AES_k\left(r+1\right) \oplus m_1\right)\left(AES_k\left(r+2\right) \oplus m_l\right) \dots \left(AES_k\left(r+l\right) \oplus m_l\right)$$

This is called *counter mode*

Theorem 12. *If F is a PRF, then counter mode is CPA secure.*

Proof overview. Assume for simplicity that all messages consist of l blocks. The sequence

$$s_i = \left(r_i, F_k\left(r_i+1\right), \dots, F_k\left(r_i+l\right)\right)$$

used for encrypting the ith message is pseudorandom. Any s_i, and $s_{i'}$ are "independent" , unless $r_i + j = r_{i'} + j'$ □

There is also ECB mode (electronic code book), where

$$Enc_k\left(m_1 \dots m_l\right) = \left(F_k\left(m_1\right), F_k\left(m_2\right), \dots, F_k\left(m_l\right)\right)$$

ECB mode is deterministic and thus **not secure**.

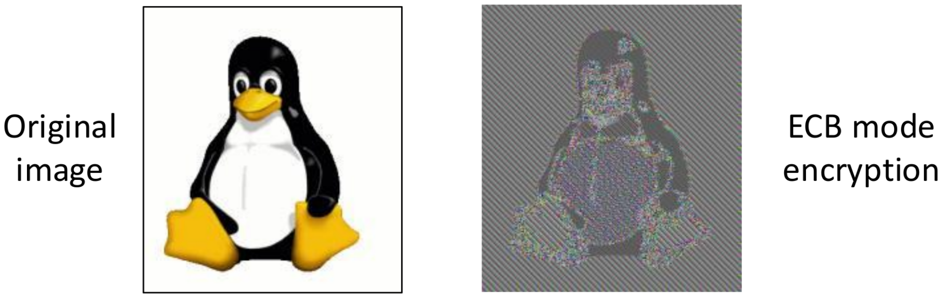


Figure 1: ECB mode

We can map from PRGs to PRFs

Theorem 13. *Let G be a PRG with expansion $2n$, then there exists a PRF F mapping n-bit inputs to n-bit outputs*

We will construct as follows: We will denote $G\left(s\right) = G_0\left(s\right)G_1\left(s\right)$, where

$$\left|G_0\left(s\right)\right| = \left|G_1\left(s\right)\right| = \left|s\right|$$

We will define

$$F_k\left(x\right) = G_{x_n}\left(\dots G_{x_1}\left(k\right)\dots\right)$$
$$\text{where } x = x_1 \dots x_n$$

We can then construct a binomial tree, starting at k, who has the children $G_0\left(k\right)$ and $G_1\left(k\right)$, and keep this iteration going. The series of subfixes to G will indicate the value given to F_k, for example the node $G_1\left(G_1\left(k\right)\right)$ will represent $F_k\left(11\right)$.

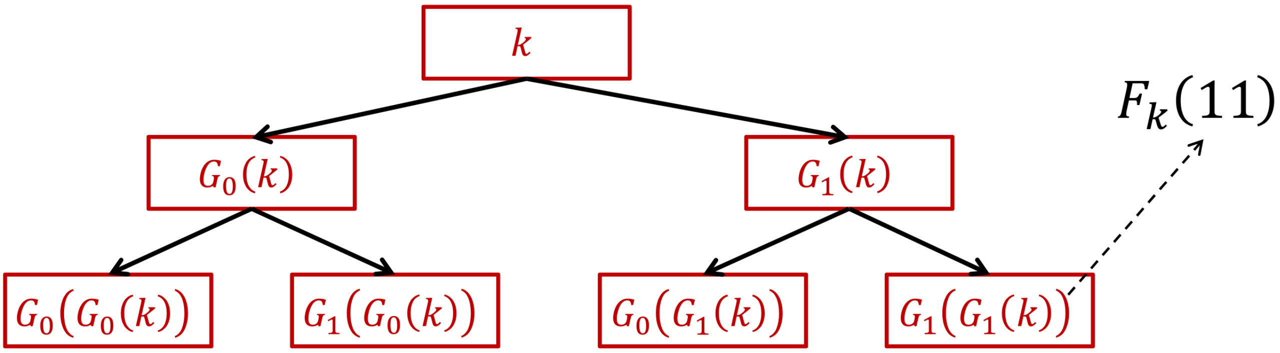


Figure 2: PRG to PRF tree

This can be proven using the hybrid $\mathcal{H}_i$. Values on level i are sampled uniformly, and independently. Values on all levels $\geq i$ are computed using G. This has the problem that level i has 2^i values, but we can resolve this given $\mathcal{D}$ runs in time $t = t\left(n\right)$, we need at most t values for the ith level. We reduce to distinguishing $G\left(s_1\right) \dots G\left(s_t\right)$ from U_{2tn}

Part IV

Tutorial 2 — 2025-11-05

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

20 Recap

So far, we have discussed *perfect secrecy*, and demonstrated its requirements, along with methods such as the one time pad. It came with the significant drawbacks of each key having to be single use, and it needing exceedingly large keys (at least the length of the plaintext). We went on to discuss a slight reduction of the security provided by this with *indistinguishable encryption* (IND), which makes use of pseudo random generators, that take a smaller key, and output a much larger pseudo random output. This can be used with a one time pad, thus allowing shorter keys, but each key can still only be used once.
We also defined *semantic security*, a powerful mathematical definition that should be reviewed in last weeks notes. We also defined *one way functions*, which we will not use extensively, but may well return to come the end of the semester.

21 Pseudorandom Generators (PRGs)

Definition 21.1 (PRG)**.** *A PRG is a polytime computable function $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ for $l(n) > n$.*

PRGs enable that for every PPT D, there exists $\mathcal{V}$, a negligible function $\forall n$ such that

$$\left| \Pr_{s \leftarrow \{0,1\}^n} \left[D\left(G\left(s\right)\right) = 1 \right] - \Pr_{r \leftarrow \{0,1\}^{l(n)}} \left[D\left(r\right) = 1 \right] \right| < \mathcal{V}\left(n\right)$$

Exercise 4. *Given a PRG G, such that $\forall n \in \mathbb{N},\ s \in \{0,1\}^n$ it holds that $G(s) \in \{0,1\}^{l(n)}$ for some function $l : l(n) > n$. Show that there exists a negligible function $\mathcal{V}(\cdot)$ such that*

$$\frac{1}{8} - \mathcal{V}\left(n\right) \leq \Pr_{s \leftarrow \{0,1\}^n} \left[G\left(s\right) \textit{ starts with } 000 \right] \leq \frac{1}{8} + \mathcal{V}\left(n\right)$$

Solution. We prove by contradiction. Assume there exists a polynomial $p(\cdot)$ such that for infinitely many n,

$$\Pr_{s \leftarrow \{0,1\}^n} \left[G\left(s\right) \text{ starts with } 000 \right] > \frac{1}{8} + \frac{1}{p\left(n\right)}$$

or

$$\Pr_{s \leftarrow \{0,1\}^n} \left[G\left(s\right) \text{ starts with } 000 \right] < \frac{1}{8} - \frac{1}{p\left(n\right)}$$

Without loss of generality, we will assume that the first inequality is true, for $p(\cdot)$, and infinite ns. We will construct a distinguisher D, such that

$$D\left(y \in \{0,1\}^*\right) = \begin{cases} 1, & \text{if } y \text{ starts with } 000 \\ 0, & \text{else} \end{cases}$$

We will note that D is a PPT. Since in a string of length $l(n)$, each bit is sampled randomly from $\{0,1\}$, we know that

$$\Pr_{r \leftarrow \{0,1\}^{l(n)}} \left[r \text{ starts with } 000 \right] = \frac{1}{2^3} = \frac{1}{8}$$

Now, D's distinguishing advantage between random strings, and the output of G may be calculated as follows: By the definition of D, and according to the contradicting assumption, for enough values of $n \in \mathbb{N}$ it holds that:

$$\begin{aligned} \left| \Pr_{s \leftarrow \{0,1\}^n} \left[D\left(G\left(s\right)\right) = 1 \right] - \underset{r \leftarrow \{0,1\}}{\overset{l(n)}{\Pr}} \left[D\left(r\right) = 1 \right] \right| &> \left| \frac{1}{8} + \frac{1}{p\left(n\right)} - \frac{1}{8} \right| \\ &\geq \frac{1}{p\left(n\right)} \end{aligned}$$

which contradicts the assumption that G is a PRG. □

We will note that in this proof we used 2 facts that are related to the start of the string beginning with 000:

1. This may be checked efficiently

2. This property has a probability of $\frac{1}{8}$ of occurring, for a string that is chosen from a uniform distribution

Conclusion: Every feature on binary strings that can be efficiently tested exists for the output of a PRG with approximately the same probability as it exists in a string sampled from the uniform distribution (the difference between the probabilities is bounded b a negligible function).

Exercise 5. *Let there be G_1, G_2 PRGs. We will define*

$$G(s) = G_1(s) || G_2(s)$$

Where the double line is string joining. Prove or disprove that G is a PRG.

Solution. This is not the case. Consider the case where $G_1 = G_2$. In this case, the output of G will not be pseudorandom, since we can construct a distinguisher that checks if the first half of the string is equal to the second half: More formally, let there be $H = G_1 = G_2$. Therefore, $G(s) = H(s) || H(s)$. We can theorise that for all H, G is not a PRG. Let us construct $D(z)$:

1. Check if the first and second half are the same.
2. If yes, return 1
3. Else, return 0

For G,

$$\Pr[D(G(s)) = 1] = 1$$

However,

$$\Pr[D(s) = 1] = \frac{1}{2^{\frac{n}{2}}}$$

As a result, G is **not** a PRG. □

22 Indistinguishable proofs

Let there be $\Pi = (KeyGen, Enc, Dec)$, an encryption system. For every algorithm A, and $\forall n \in \mathbb{N}$, we will define the experiment $IND_{\Pi,A}(n)$, as follows:

1. $k \leftarrow KeyGen(1^n)$
2. A receives 1^n as input, and outputs a pair of messages (m_0, m_1)
3. $b \leftarrow \{0,1\}$, and we compute $c^* \leftarrow Enc_k(m_b)$, and pass c^* to A
4. A returns $b' \in \{0,1\}$

$IND_{\Pi,A}(n) = 1$ if $b' = b$. Otherwise, $IND_{\Pi,A}(n) = 0$. We will say that Π is an indistinguishable encryption system if for every PPT function A, there exists a negligible function $v(\cdot)$, such that

$$\Pr[IND_{\Pi,A}(n) = 1] \leq \frac{1}{2} + v(n)$$

for every $n \in \mathbb{N}$.

Exercise 6. *Let there be $l(n) : \forall n \in \mathbb{N}\ l(n) > n$. Let there be a determinstic PT G, such that for all $n \in \mathbb{N}$, and for all $s \in \{0,1\}^n$, it holds that $G(s) \in \{0,1\}^{l(n)}$. We will consider the system $\Pi = (KeyGen, Enc, Dec)$, defined as follows:*

1. $k \leftarrow KeyGen(1^n)$
2. $Enc_k(m) \implies c = G(k) \oplus m$
3. $Dec_k(c) \implies m = c \oplus G(k)$

It is given that Π is IND-secure. Is G necessarily a PRG?

Solution. G is necessarily a PRG. Let us assume the contradiction that it is not: There exists a PPT D, and there exists the polynomial $p(\cdot)$, such that

$$\left| \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{l(n)}}[D(r) = 1] \right| > \frac{1}{p(n)}$$

In this case, it will hold that Π is not IND-secure. Let us assume that there are infinite values of n, that enable the above inequality, but without the absolute value:

$$\Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{l(n)}}[D(r) = 1] > \frac{1}{p(n)}$$

Let us consider the algorithm A that partakes in the experiment $IND_{\Pi,A}(n)$, defined as follows:

1. A generates $m_0 \leftarrow \{0,1\}^{l(n)}$, and $m_1 \leftarrow 0^{l(n)}$.
2. A receives c^*, for m_b, and runs $D(c^*)$, and returns 1 **if and only if** D returns 1

When A returns 0, then it is also true that $c^* = G(k) \oplus m_0$ is distributed evenly over $\{0,1\}^{l(n)}$. Therefore:

$$\Pr[IND_{\Pi,A}(n) = 1 | b = 0] = 1 - \Pr_{r \leftarrow \{0,1\}^{l(n)}}[D(r) = 1]$$

Therefore, when A returns 1, it holds that $c^* = G(k) \oplus 0^{l(n)} = G(k)$, where $k \leftarrow \{0,1\}^n$. Therefore

$$\Pr[IND_{\Pi,A}(n) = 1 | b = 1] = \Pr_{s \leftarrow \{0,1\}^n}[D(r) = 1]$$

Overall, we get for infinite values of $n \in \mathbb{N}$, it holds that

$$\begin{aligned}\Pr[IND_{\Pi,A}(n) = 1] &= \Pr[IND_{\Pi,A}(n) = 1 | b = 0] \cdot \Pr[b = 0] + \Pr[IND_{\Pi,A}(n) = 1 | b = 1] \cdot \Pr[b = 1] \\ &= \Pr[D(G(\text{PRG})) = 1] \cdot \frac{1}{2} + (1 - \Pr[D(\text{random}) = 0]) \cdot \frac{1}{2} \\ &= \frac{1}{2}\left(\Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] + 1 - \Pr_{r \leftarrow \{0,1\}^{l(n)}}[D(r) = 1]\right) \\ &\geq \frac{1}{2} + \frac{1}{2p(n)} \\ &> \frac{1}{2} + \frac{1}{p'(n)}\end{aligned}$$

The final inequalities follows from our initial assumption:

$$\Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{l(n)}}[D(r) = 1] > \frac{1}{p(n)}$$

This is a contradiction to the given that Π is IND-secure. Therefore, G is a PRG. If we consider the other direction for our assumption (because of the absolute value), as in:

$$\Pr_{r \leftarrow \{0,1\}^{l(n)}}[D(r) = 1] - \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] > \frac{1}{p(n)}$$

Then we may use the same proof, but swapping A to return the opposite of D, i.e., D returns 0, A returns 1. □

Exercise 7. *Let there be $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$, efficiently computable distributions that are computationally indistinguishable.*

1. *Prove that for every PT function f, the distributions $f(Y) = \{f(Y_n)\}_{n \in \mathbb{N}}$ and $f(X) = \{f(X_n)\}_{n \in \mathbb{N}}$ are computationally indistinguishable.*
2. *Is the theorem in part 1 still true if f is not PT?*

Solution. 1. We will assume the contradiction that there exists a PPT function f, for which the distributions $f(X)$ and $f(Y)$ are distinguishable. Then, there exists a PPT distinguisher A such that for infinitely many n:

$$\left|\Pr_{x \leftarrow X_n}[A(1^n, f(x)) = 1] - \Pr_{y \leftarrow Y_n}[A(1^n, f(y)) = 1]\right| > \frac{1}{p(n)}.$$

Let us define a new distinguisher $D(1^n, z) = A(1^n, f(z))$. We will note that since A is a PPT, then so too is D. Therefore, for every $n \in \mathbb{N}$ it holds that:

$$\Pr_{x \leftarrow X_n}[D(1^n, x)] = \Pr_{x \leftarrow X_n}[A(1^n, f(x)) = 1]$$

and also

$$\Pr_{y \leftarrow Y_n}[D(1^n, y)] = \Pr_{y \leftarrow Y_n}[A(1^n, f(y)) = 1]$$

Therefore, for infinite values of $n \in \mathbb{N}$

$$\left|\Pr_{x \leftarrow X_n}[D(1^n, x) = 1] - \Pr_{y \leftarrow Y}[D(1^n, y) = 1]\right| > \frac{1}{p(n)}$$

So D is a PPT algorithm, that distinguishes between X and Y, with non negligible probability for infinite values of $n \in \mathbb{N}$, which is a contradiction to the assumption that these two distributions are indistinguishable.

2. It is not. Let there be G, a PRG, such that for every $n \in \mathbb{N}$, $G : \{0,1\}^n \to \{0,1\}^{n+1}$. We will look at the following example:

- $\forall n \in \mathbb{N}$ the distribution X_n takes a random $s \leftarrow \{0,1\}^n$, and returns $G(s)$
- $\forall n \in \mathbb{N}$ the distribution X_n takes a random $r \leftarrow \{0,1\}^{n+1}$, and returns r
- $\forall n \in \mathbb{N}$, given the input $y \in \{0,1\}^{n+1}$, the function f checks if y is in the image of G (as in, if there exists $s \in \{0,1\}^n : y = G(s)$). If so, $f(y) = 1$, otherwise $f(y) = 0$

Firstly, we will note that since G is a PRG, it holds that the distributions $X = \{X_n\}_{n\in\mathbb{N}}$, and $Y = \{Y\}_{n\in\mathbb{N}}$ are indistinguishable. Now, we will show that $f(X)$ and $f(Y)$ are distinguishable. Let there be D, an algorithm that given input $(1^n, b)$, where $b \in \{0,1\}$, returns the bit b. For every $n \in \mathbb{N}$ it holds that:

$$\left|\Pr_{x\leftarrow X_n}\left[D\left(1^n, f(x)\right) = 1\right] - \Pr_{y\leftarrow Y_n}\left[D\left(1^n, f(y) = 1\right)\right]\right|$$

$$= \left|\Pr_{s\leftarrow\{0,1\}^n}\left[D\left(1^n, f\left(G(s)\right)\right) = 1\right] - \Pr_{r\leftarrow\{0,1\}^{n+1}}\left[D\left(1^n, f(r)\right) = 1\right]\right|$$

$$= \Bigg|\Pr_{s\leftarrow\{0,1\}^{n+1}}[D(1^n, 1) = 1] - \Pr_{r\leftarrow\{0,1\}^{n+1}}[r \in \text{Image}(G)] \cdot \Pr_{r\leftarrow\{0,1\}^{n+1}}[D(1^n, f(r)) = 1 \mid r \in \text{Image}(G)]$$

$$- \Pr_{r\leftarrow\{0,1\}^{n+1}}[r \notin \text{Image}(G)] \cdot \Pr_{r\leftarrow\{0,1\}^{n+1}}[D(1^n, f(r)) = 1 \mid r \notin \text{Image}(G)]\Bigg|$$

$$= \left|1 - \Pr_{r\leftarrow\{0,1\}^{n+1}}\left[r \in Image\left(G\right)\right] \cdot 1 - \Pr_{r\leftarrow\{0,1\}^{n+1}}\left[r \notin Image\left(G\right)\right] \cdot 0\right|$$

$$= \left|1 - \frac{|Image\left(G\right)|}{2^{n+1}}\right|$$

$$\geq \frac{1}{2}$$

□

Part V
Lecture 4 — 2025-11-19

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

23 Introduction

We are going to discuss Message Authentication Codes, where we verify the authenticity of a sent message. You receive a message, and can verify that it came from *me*, and nobody else (Like Eve. Seriously, screw Eve. Or don't, that might be what she wants).

We are going to discuss authenticating fixed length messages, and arbitrary length messages. We will do this through collision resistant hash functions, where we use the "hash and authenticate" paradigm. After this, we will link it back to encryption.

24 Message authentication

Alice and Bob wish to communicate, and Eve completely controls the channel. We would like to assure the receiver of a message, that it has not been modified. If Alice sends Bob a message that says "Pay Charlie $10", Eve can change it to say "Pay Eve $10,000" (You may also pretend that Bob is your Bank). Encryption ensures data secrecy, that no one else knows the contents, and *authentication* ensures the integrity of the data, that it remained unchanged. These concepts are orthogonal, one does not enable the other.

24.1 Message Authentication Code (MAC)

The syntax is $\Pi = (Gen, Mac, \mathrm{Vrfy})$, where the key-generation algorithm Gen on input 1^n outputs a key k, the tag generation algorithm Mac takes a key k, and a message $m \in \{0,1\}^*$, and outputs a tag $t \in \{0,1\}^*$, and the verification algorithm Vrfy takes a key k, message m, and tag t, and outputs a bit b.
The correctness comes from:

$$\forall k, m\ \mathrm{Vrfy}_k\left(m, Mac_k\left(m\right)\right) = 1$$

The security of MACs is found as follows: The adversary $\mathcal{A}$ can adaptively ask for tags of messages of its choice, and attempts to forge a valid tag on a new message (m^*, t^*). Let Q be the set of all queries asked by $\mathcal{A}$. We then have

$$MacForge_{\Pi,\mathcal{A}}\left(n\right) = \begin{cases} 1, & \text{if } \mathrm{Vrfy}_k\left(m^*, t^*\right) = 1 \wedge m^* \notin Q \\ 0, & \text{otherwise} \end{cases}$$

Definition 24.1 (MAC scheme)**.** *A MAC scheme $\Pi = (Gen, Mac, Vrfy)$ is secure if for every PPT adversary $\mathcal{A}$, there exists a negligible function $v(\cdot)$ such that*

$$\Pr\left[MacForge_{\Pi,\mathcal{A}}\left(n\right) = 1\right] \leq v\left(n\right)$$

This definition does **not** prevent replay attacks. Eve may send the message "Send Charlie $10" as many times as she likes, and it will appear legitimate, thanks to the deterministic MAC.

24.2 Fixed length MAC

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a PRF.

- Key generation: Sample $k \leftarrow \{0,1\}^n$
- Tag generation: On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$ output $t = F_k\left(m\right)$
- Verification: On input $k \in \{0,1\}^n$, $m \in \{0,1\}^n$, and $t \in \{0,1\}^n$, output 1 if $t = F_k\left(m\right)$, and 0 otherwise

Theorem 14. *If F is a PRF, then the above MAC scheme is secure*

Proof . The concept is that given a forger $\mathcal{A}$, for the MAC scheme, we construct a distinguisher $\mathcal{D}$ for the PRF. $\mathcal{D}$ has oracle access to a function $\mathcal{O}$, which is either F_k, or a truly random h. Additionally, $\mathcal{D}$ runs $\mathcal{A}$ internally, and simulates the experiment $MacForge_{\Pi,\mathcal{A}}$ to $\mathcal{A}$ using $\mathcal{O}$.

Let us assume towards a contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p(n)$ such that

$$\Pr\left[MacForge_{\Pi,\mathcal{A}}\left(n\right) = 1\right] \geq \frac{1}{p\left(n\right)}$$

for infinitely many ns. The distinguisher $\mathcal{D}^{\mathcal{O}}$ will invoke $\mathcal{A}$, and respond to each of its queries m with $t = \mathcal{O}\left(m\right)$. It will output 1 **if and only if** $m^* \notin \mathcal{Q}$, and $t^* = \mathcal{O}\left(m^*\right)$, where $\mathcal{Q}$ is the set of all queries asked by $\mathcal{A}$. There are 2

cases:
Case 1: If $\mathcal{O} = F_k$ is a PRF, then $\mathcal{A}$'s view is identical to $MacForge_{\Pi,\mathcal{A}}(n)$, and so

$$\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] = \Pr\left[MacForge_{\Pi,\mathcal{A}}(n) = 1\right]$$

Case 2: If $\mathcal{O} = h$ is a truly random function, then if $m^* \notin \mathcal{Q}$, then $\mathcal{A}$'s view is independent of $\mathcal{O}(m^*)$, and so

$$\Pr\left[D^{h(\cdot)}(1^n) = 1\right] = 2^{-n}$$

From here, we may calculate

$$\begin{aligned}
&\left|\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[D^{h(\cdot)}(1^n) = 1\right]\right| && (1)\\
&= \left|\Pr\left[MacForge_{\Pi,\mathcal{A}}(n) = 1\right] - 2^{-n}\right| && (2)\\
&\geq \frac{1}{p(n)} - 2^{-n} && (3)\\
& && (4)
\end{aligned}$$

So, we have successfully distinguished between a truly random function, and a PRF, which is a contradiction. □

24.3 Arbitrary length messages

Given

$$\widehat{\Pi} = \left(\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Vrfy}}\right)$$

for fixed length messages, we want to define $\Pi = (Gen, Mac, \text{Vrfy})$ for arbitrary length messages. Here is a first (naïve) attempt:

24.3.1 Attempt 1

Let $Mac_k(m) = (t_1, \ldots, t_d)$ where $t_i = \widehat{\text{Mac}}_k(m_i)$, $Gen = \widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1, \ldots, m_d), (t_1, \ldots, t_d)) = 1$ **if and only if** $\widehat{\text{Vrfy}}_k(m_i, t_i) = 1$ for every $i \in [d]$.

This is completely insecure. Consider

$$t = (t_1, t_2)$$

If t is avalid tag for $m = (m_1, m_2)$, then it also holds that $t^* = t_1$ is a valid tag for $m^* = m_1$.

24.3.2 Attempt 2

$Mac_k(m) = (t_1, \ldots, t_d)$, where $t_i = \widehat{\text{Mac}}_k(d, m_i)$, $\text{Gen} = \widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1, \ldots, m_d), (t_1, \ldots, t_d)) = 1$ **if and only if** $\widehat{\text{Vrfy}}_k((d, m_i), t_i) = 1$ for every $i \in [d]$.

This is also insecure. Consider if $t = (t_1, t_2)$ is a valid tag for $m = (m_1, m_2)$, then $t^* = (t_2, t_1)$ is a valid tag for $m^* = (m_2, m_1)$.

24.3.3 Attempt 3

$Mac_k(m) = (t_1, \ldots, t_d)$, where $t_i = \widehat{\text{Mac}}_k(d, i, m_i)$, $\text{Gen} = \widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1, \ldots, m_d), (t_1, \ldots, t_d)) = 1$ **if and only if** $\widehat{\text{Vrfy}}_k((d, i, m_i), t_i) = 1$ for every $i \in [d]$.

This is still insecure. If $t = (t_1, t_2)$ is a valid tag for $m = (m_1, m_2)$, and similarly $t' = (t'_1, t'_2)$ is a valid tag for $m' = (m'_1, m'_2)$, then $t^* = (t_1, t'_2)$ is a valid tag for $m^* = (m_1, m'_2)$

24.3.4 Solution 1

$Mac_k(m) = (r, t_1, \ldots, t_d)$, where $t_i = \widehat{\text{Mac}}_k(r, d, i, m_i)$, and r is sampled uniformly, and independently for each message m. $\text{Gen} = \widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1, \ldots, m_d), (r, t_1, \ldots, t_d)) = 1$ **if and only if** $\widehat{\text{Vrfy}}_k((r, d, i, m_i), t_i) = 1$ for every $i \in [d]$.

This is a solution, but has the drawback of very long tags.

24.3.5 Solution 2 (CBC-MAC)

Let $Mac_k(m) = t_d$, where

- $t_0 = 0^n$, and $t_i = F_k(t_{i-1} \oplus m_i)$ for $i \in [d]$
- F_k can be any PRF
- d must be fixed ahead of time

In short, we take the result of the signature of the first block m_1, and XOR it with the second block m_2, and then compute the signature of that. This is continued on recursively until the end:

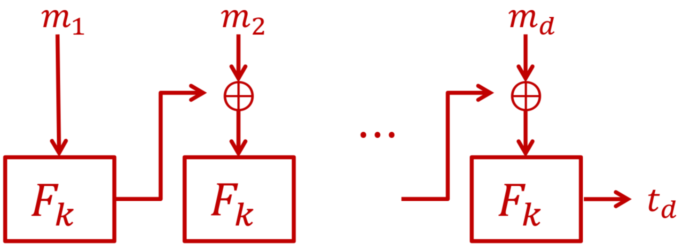


Figure 3: CBC-MAC

This is great because it is very easy to implement, so that solves many bug issues, and lots of the parts may be implemented in hardware.

24.3.6 Solution 3 - Hash and Authenticate

We will compress m into a short fingerprint $H(m)$, and then authenticate $H(m)$ instead of m itself. It is critical for our hashing function H, that it is very difficult to find $m \neq m'$ such that $H(m) = H(m')$.

25 Collision-Resistant Hash Functions

The purpose of Collision-Resistant Hash Functions is that they compress arbitrarily long inputs into short, fixed-length outputs. It should be very hard to find $x \neq x'$ such that $H(x) = H(x')$.

Syntactically: $\Phi = (\text{Gen}, H)$:

- The key generation algorithm Gen, on input 1^n outputs a key s
- The evaluation algorithm H on input s, and $x \in \{0,1\}^*$ outputs $H_s(x) \in \{0,1\}^{l(n)}$

So, our adversary is playing

$$HashColl_{\Phi,\mathcal{A}}(n) = \begin{cases} 1, & \text{if } H_s(x) = H_s(x') \wedge x \neq x' \\ 0, & \text{otherwise} \end{cases}$$

Bringing us to the definition

Definition 25.1 (Collision Resistant)**.** *Φ is* ***collision resistant*** *if for every PPT adversary $\mathcal{A}$ there exists a negligible function $v(\cdot)$ such that*

$$\Pr\left[HashColl_{\Phi,\mathcal{A}}(n) = 1\right] \leq v(n)$$

Can we find collisions? Well, if our function produces output of length n bits, then we could $2^n + 1$ inputs, but this will take forever. Instead, we have **The Birthday Attack**.

Given $H : \{0,1\}^* \to \{0,1\}^l$, sample $q = \mathcal{O}\left(2^{\frac{l}{2}}\right)$ inputs uniformly and independently. This finds colliding pair of inputs with a constant probability (2^l pairs). To think about this in general, given output of 2^n, then similar to the birthday paradox from probability and statistics, taking $\sqrt{2^n} = 2^{\frac{n}{2}}$ possible inputs, and testing them, will give us a shared output with probability of 50%.

In practice $l \geq 128$. Popular heuristic (unkeyed) functions include MD5, SHA1, SHA3, and so on. Both MD5, and SHA1 are horrifically insecure, but still heavily used. They can still be useful in non crypto contexts, but should now *never* be used in a crypto context.

Generally, in order to to create an arbitrary length hash function, we start with a fixed length has function, and extend it.

26 Authenticating Arbitrary-Length Messages

Given

$$\hat{\Pi} = \left(\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Vrfy}}\right)$$

for fixed length messages, and a collision resistant hash function $\Phi = (Gen_H, H)$, define $\Pi = (Gen, Mac, \text{Vrfy})$ for arbitrary length messages.

Reconsider Solution 3 above:

- $Mac_{k,s}(m) = \widehat{Mac}_k(H_s(m))$

- $Gen = \left(\widehat{Gen}, Gen_H\right)$

- $\text{Vrfy}_{k,s}(m,t) = 1 \Leftrightarrow \widehat{Vrfy}_k(H_s(m),t) = 1$

Exercise 8. *Let $Mac_k(m)$ samples $s \leftarrow Gen_H(1^n)$, and outputs $\left(s, \widehat{Mac}_k(H_s(m))\right)$. Is this secure?*

Solution. This is insecure. In general, our family our functions should overall be difficult to find collisions within, but it is *possible* that there exists a function within the family where it is incredibly easy to find within it collisions. Since s is generated, our adversary can choose an s for which the function H_s has many collisions. Consider the following:

- The adversary requests the signature of 0, and gets in response $(s, Mac_k(H_s(0)))$. Let us denote $a = H_s(0)$.

- We will now find a function within the family that maps *everything* to a.

- The adversary will now pick this value for s, and the signature of every message m will be

$$Mac_k(H_s(m)) = Mac_k(a) = Mac_k(H_s(0))$$

□

Let us return to Solution 3.

Theorem 15. *If $\widehat{\Pi}$ is a secure MAC, and Φ is collision resistant, then Π is a secure MAC.*

Proof . Consider the event "collision". $\mathcal{A}$ asks for a tag on some m_i such that $m_i \neq m^*$, and $H_s(m_i) = H)s(m^*)$. Whenever "collision" occurs, then we can use $\mathcal{A}$ to find a non trivial collision, and whenever "collision" does not occur (and $\mathcal{A}$ wins), then we can use $\mathcal{A}$ to forge a tag on the fixed length message

$$H_s(m^*) \notin \{H_s(m_1), \ldots, H_s(m_q)\}$$

Let $\mathcal{A}$ be any PPT adversary, then

$$\Pr[MacForge_{\Pi,\mathcal{A}}(n) = 1] \leq \Pr[\text{collision}] + \Pr\left[MacForge_{\Pi,\mathcal{A}}(n) \wedge \overline{\text{collision}}\right]$$

Since

$$\Pr[A] = \Pr[A \wedge B] + \Pr\left[A \wedge \overline{B}\right] \leq \Pr[B] + \Pr\left[A \wedge \overline{B}\right]$$

Let **claim I** be that there exists a negligible function $v_1(n)$ such that

$$\Pr[\text{collision}] \leq v_1(n)$$

and **claim II** be that there exists a negligible function $v_2(n)$ such that

$$\Pr\left[MacForge_{\Pi,\mathcal{A}}(n) \wedge \overline{\text{collision}}\right] \leq v_2(n)$$

Proof of Claim I:
Let us assume towards a contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p(n)$ such that

$$\Pr[\text{collision}] \geq \frac{1}{p(n)}$$

The collision finder $\mathcal{C}$ will

- On input s, sample $k \leftarrow \widehat{Gen}(1^n)$, and invoke $\mathcal{A}$

- Respond to $\mathcal{A}$'s queries, using (k, s)

Proof of Claim II:
Let us assume towradrs contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p(n)$, such that

$$\Pr\left[MacForge_{\Phi,\mathcal{A}}(n) = 1 \wedge \overline{\text{collision}}\right] \geq \frac{1}{p(n)}$$

for infinitely many ns. The forger $\hat{\mathcal{A}}$ will

- Sample $s \leftarrow Gen_H(1^n)$, and invoke $\mathcal{A}$

- Respond to $\mathcal{A}$'s queries m_i by forwarding $H_s(m_i)$ to the oracle $\widehat{Mac}_k(\cdot)$, and then forwarding back its response to $\mathcal{A}$

- Output $(H_s(m^*), t^*)$

So

$$\Pr\left[MacForge_{\hat{\Pi},\hat{\mathcal{A}}}(n) = 1\right] = \Pr\left[MacForge_{\Pi,\mathcal{A}}(n) = 1 \wedge \overline{\text{collision}}\right]$$
$$\geq \frac{1}{p(n)}$$

Which is a contradiction. □

27 Returning to encryption

Recall that CPA-secure encryptions may be built from any PRF, and hold that

$$Enc_k(m; r) = (r, F_k(r) \oplus m)$$

This has a rather serious problem. An adversary can change the contents of a message, without the recipient knowing. The adversary will also not know how he has changed it, but he can change it, and none will know that this has happened.

$$Enc_k(m \oplus 1^n; r) = (r, F_k(r) \oplus m \oplus 1^n)$$

So, we want to achieve both encryption, and authenticated messages, creating **authenticated encryption**.

Let us consider

$$k = (k_E, k_M) \tag{5}$$
$$c \leftarrow Enc_{k_E}(m) \tag{6}$$
$$t \leftarrow Mac_{k_M}(m, t) \tag{7}$$

Which can then all be reversed

$$k = (k_E, k_M) \tag{8}$$
$$m \leftarrow Dec_{k_E}(c) \tag{9}$$
$$? \leftarrow Mac_{k_M}(m, t) \tag{10}$$

This may well be insecure. We have no guarantees that our MAC system does not leak information about the message. In fact, we may completely leak m, since we have no security guarantees.

This may be improved to

$$k = (k_E, k_M) \tag{11}$$
$$c \leftarrow Enc_{k_E}(m) \tag{12}$$
$$t \leftarrow Mac_{k_M}(c, t) \tag{13}$$

Which can then all be reversed

$$k = (k_E, k_M) \tag{14}$$
$$m \leftarrow Dec_{k_E}(c) \tag{15}$$
$$? \leftarrow Mac_{k_M}(c, t) \tag{16}$$

If the encryption is CPA-secure, and the MAC is secure, then this construction is a CPA-secure encryption scheme, with a secure MAC.

27.1 Chosen Ciphertext Attack CCA

This has brought us to chosen ciphertext attack schemes. Here, $\mathcal{A}$ can adaptively ask for encryptions of messages of its choice, and for decryptions of ciphertexts of its choice (aside from the challenge ciphertext c^*).

Definition 27.1 (CCA-IND)**.** Π *has indistinguishable encryptions under a chosen-ciphertext attack if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $v(\cdot)$ *such that*

$$\Pr\left[IND^{CCA}_{\Pi,\mathcal{A}}(n) = 1\right] \leq \frac{1}{2} + v(n)$$

In this case, we may also say that Π *is CCA-secure.*

We will observe that CCA-security implies authenticity. Given $Enc_k(m)$, it is hard to generate $Enc_k(m')$ for a "related" m' (such as $m' = m+1$). This is because if he could, then he could take c^*, and flip the bits. This is (under the assumption) a legal encryption, which could then be decrypted into the inverse bits of m_b, so he could then take the decryption of the inverse of c^*, flip its bits, and identify which m_i was returned encrypted to c^*.

We will note that it feels a little unrealistic. Honest parties do not typically decrypt arbitrary adversarially chosen ciphertexts. Nevertheless, adversaries may be able to influence what gets encrypted / decrypted, and learn some partial information. For example, in WWII, the US cryptanalysts might have tried to send encrypted messages to the Japananese, and then monitor their behaviour. An adversary may send certain ciphertexts on behalf of a user to the user's bank, and then the bank will decrypt these ciphertexts and its response will leak information to the adversary. Furthermore, an encryption scheme might be used as part of an authentication protocol where one party sends a ciphertext to the other, who then decrypts it and returns the result. Thus we can see, that CCA is a realistic security requirement in the real world.

We will thus note that CPA security does **not** imply CCA security, since we can affect the output of encryption function, and get another valid encryption:

$$Enc_k(m; r) = (r, F_k(r) \oplus m)$$
$$Enc_k(m \oplus 1^n; r) = (r, F_k(r) \oplus m \oplus 1^n)$$

27.1.1 CCA-Secure encryption scheme

The main idea here is that adversaries should not be able to generate new "valid" ciphertexts, and that the decryption oracle becomes useless. Our solution is called *Encrypt-then-Authenticate*:
Let $\Pi_E = (KeyGen_E, Enc, Dec)$ be a CPA secure encryption scheeme, and let $\Pi_M = (KeyGen_M, Enc, Dec)$ be a secure MAC. Let us define $\Pi' = (KeyGen', Enc', Dec')$ as

- $KeyGen'(1^n)$ output $k = (k_E, k_M)$ where $k_E \leftarrow Gen_E(1^n)$ and $k_M \leftarrow KeyGen_M(1^n)$
- $Enc'_k(m)$ output (c, t) where $c \leftarrow Enc_{k_E}(m)$ and $t \leftarrow Mac_{k_M}(c)$
- $Dec'_k(c, t)$ If $Vrfy_{k_M}(c, t) = 1$ then output $Dec_{k_E}(c)$, otherwise output $\perp$

Theorem 16. *Let us assume that Π_E is a CPA-secure encryption scheme, and that Π_M is a secure MAC with* ***unique tags****, then Π' is a CCA secure encryption scheme.*

Proof . We will first note that MAC with unique tags means that for any key k_M and message m, there exists *exactly* one t such that $Vrfy_{k_M}(m, t) = 1$. Any MAC scheme with a deterministic Mac algorithm can be modified to have unique tags by using "canonical verification" instead of its own Vrfy algorithm: On input (k_M, m, t) output 1 if and only if $t = Mac_{k_M}(m)$ and 0 otherwise

Proof idea:
We will call (c, t) valid w.r.t (k_E, k_M) if $Vrfy_{k_M}(c, t) = 1$, and invalid otherwise. Consider the event Valid-Query. $\mathcal{A}$ queries the decryption oracle with a valid ciphertext, that was **not** produced by the encryption oracle. If $\Pr[ValidQuery]$ is non-negligible, then we can use $\mathcal{A}$ to break the MAC Π_M. If it is negligible, then we can use $\mathcal{A}$ to break the CPA security of Π_E

Let $\mathcal{A}$ be a PPT adversary, then:

$$\Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n) = 1\right] \tag{17}$$
$$\leq \Pr[ValidQuery] + \Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n) = 1 \wedge \overline{ValidQuery}\right] \tag{18}$$

Theorem 17 (Claim II)**.** ***Claim II****: There exists a negligible $v(n)$ such that*

$$\Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n) = 1 \wedge \overline{ValidQuery}\right] \leq \frac{1}{2} + v(n)$$

Proof . We will assume towards contradiction that there exists a polynomial $p(n)$, such that for infinitely many ns

$$\Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n) = 1 \wedge \overline{ValidQuery}\right] \geq \frac{1}{2} + \frac{1}{p(n)}$$

We will construct an adversary $\mathcal{B}$ such that for infinitely many ns

$$\Pr\left[IND^{CCA}_{\Pi_E,\mathcal{B}}(n) = 1 \wedge \overline{ValidQuery}\right] \geq \frac{1}{2} + \frac{1}{p(n)}$$

$\mathcal{B}$ will sample k_M, and invoke $\mathcal{A}$, and respond to its queries. It will then output the same (m_0, m_1), and b'.
It will query the oracle $Enc_{k_E}(\cdot)$ to obtain a ciphertext c, compute $t \leftarrow Mac_{k_M}(c)$, and return (c, t).
For the decryption query, if (c, t) was a response to a previous query m, then return m, and otherwise return $\perp$.

Since ValidQuery does not occur, then $\mathcal{B}$ does not need a decryption oracle for simulating $\mathcal{A}$'s view. Thus:

$$\Pr\left[IND^{CPA}_{\Pi_E,\mathcal{B}}(n) = 1\right] \tag{19}$$
$$\geq \Pr\left[IND^{CPA}_{\Pi_E,\mathcal{B}}(n) \wedge \overline{ValidQuery}\right] \tag{20}$$
$$= \Pr\left[IND^{CPA}_{\Pi_E,\mathcal{A}}(n) \wedge \overline{ValidQuery}\right] \tag{21}$$
$$\geq \frac{1}{2} + \frac{1}{p(n)} \tag{22}$$

□

Theorem 18 (Claim I)**.** ***Claim I****: There exists a negligible $v(n)$ such that* $\Pr[ValidQuery] \leq v(n)$

Proof . Let us assume towards contradiction that there exists a polynomial $p(n)$ such that $\Pr[ValidQuery] \geq \frac{1}{p(n)}$ for infinitely many n's. Let $q(n)$ be a polynomial upper bound on the number of decryption queries made by $\mathcal{A}$. We will construct a PPT adversary $\mathcal{B}$ such that $\Pr[MacForge_{\Pi_M,\mathcal{B}} = 1] \geq \frac{1}{p(n) \cdot q(n)}$ for infinitely many ns

We will construct $\mathcal{B}$ as follows: Sample k_E, b and $i \leftarrow \{1, \ldots, q\}$, invoke $\mathcal{A}$, and respond to its queries. For an encryption query m, we will compute $c \leftarrow Enc_{k_E}(m)$, and query the oracle $Mac_{k_M}(\cdot)$ with c to obtain the tag t. Finally, return (c, t).
For the first $i - 1$ decryption queries (c, t), if (c, t) was a response to a previous encryption query m, then return m, and otherwise $\perp$. For the ith decryption query (c, t), output $(c^*, t^*) = (c, t)$ as the (potential) forgery.

We will observe that ValidQuery occurs, and that $\mathcal{B}$ geusses the index i of the first valid decryption query that was not obtained from a previous encryption query. Therefore, $Vrfy_{k_M}\left(c^*,t^*\right)=1$ and $\mathcal{B}$ did not query $Mac_{k_M}\left(\cdot\right)$ on c^*. So, we may conclude

$$\begin{aligned}\Pr\left[MacForge_{\Pi_M,\mathcal{B}}=1\right] &\geq \Pr\left[ValidQuery\wedge\mathcal{B}\text{ guesses }i\right]\\ &=\frac{\Pr\left[ValidQuery\right]}{q\left(n\right)}\\ &\geq\frac{1}{p\left(n\right)\cdot q\left(n\right)}\end{aligned}$$

□

□

28 Crypto primitives so far

So far, we have seen PRFs, PRGs, built PRGs from PRFs. From PRGs we have create IND-secure symmetric key encryption, and built CPA-secure symmetric key encryption from PRFs. Furthermore, we built IND-secure from CPA-secure. PRFs were also used to create MACs for fixed length messages, we then combined fixed length MAC and CPA-secure to make CCA secure symmetric key encryption. Finally, from MAC for fixed length messages, and Collision Resistant Hash Function, we made MAC for arbitrary length messages.

This concludes this section of the course, covering symmetric encryption.

Part VI
Tutorial 3 — 2025-11-19

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

29 Reminder

We began with One Time Pads (OTPs), that enable *perfect secrecy*, but have the problems of single use keys, and exceedingly long keys. We then moved on to *Indistinguishable Encryptions*, and *PRG*s, which enabled smaller keys. We then continued on to create a scheme, using *Pseudo Random Functions* that was resistant to *Chosen Plaintext Attacks*, and showed that PRGs are equivalent to PRFs.

Exercise 9. *Given* $\Pi_1 = (KeyGen_1, Enc_1, Dec_1)$*, and* $\Pi_2 = (KeyGen_2, Enc_2, Dec_2)$*, one of* Π_1, Π_2 *is IND. Create a scheme* Π *that is IND.*

Solution. Π will be as follows:

- $KeyGen\left(1^n\right)$: The key will be constructed of 2 parts: (k_1, k_2), where for $i \in \{1, 2\}$ k_i comes from $KeyGen_i$
- $Enc\,(k, m)$: $Enc_{k_2}\left(Enc_{k_1}(m)\right)$
- $Dec\,(k, c)$: $Dec_{k_1}\left(Dec_{k_2}(c)\right)$

The correctness of the scheme is obvious from the construction, since both the base schemes are correct. We must now prove that it is IND-secure:

Let us assume towards contradiction that the scheme is not IND-secure, so there exists an polynomial adversary A that can win the IND game against Π. There are 2 cases:

Case 1: Π_1 is IND-secure, and Π_2 is not. We will thus build the adversary B that breaks Π_1 as follows:

1. It does this by generating m_0, m_1 from A
2. $k_2 \leftarrow KeyGen_2\left(1^n\right)$
3. Get $c = Enc_{k_1}(m_b)$
4. Compute $Enc_{k_2}(c)$, and send it to A
5. A returns b', which B then returns

Case 2: Π_2 is IND-secure, and Π_1 is not. We will build B that breaks Π_2 as follows:

1. Generate m_0, m_1 from A
2. $k_1 \leftarrow KeyGen_1$
3. Create p_0, p_1 via $Enc_{k_1}(m_i)$
4. Send these for encryption by Enc_{k_2}, and then send those to A, which can by assumption differentiate them.

In both cases, we have built an adversary that can break the IND-secure scheme, which is a contradiction, and so we can conclude that Π is IND-secure. □

Exercise 10. *Let there be a PRF* $F_k, G_k : \{0,1\}^n \to \{0,1\}^n$*, which means that*

$$\forall PPT\ D : \Pr_{k \leftarrow \{0,1\}^n}\left[D^{F_k(\cdot)} = 1\right] - \Pr_{r \leftarrow Funcs_{n \to n}}\left[D^{r(\cdot)} = 1\right] < neg\,(n)$$

In short, we cannot distinguish between the output of a PRF, and a truly random function.
Prove / Disprove

$$\forall PPT\ D : \left|\Pr\left[D^{F_k(\cdot)} = 1\right] - \Pr\left[D^{G_k(\cdot)} = 1\right]\right| \leq neg\,(n)$$

Or in Spanish, para cada PPT D, no podemos distinguir entre la salida de dos PRF.
Finally, in English, sbe rirel CCG Q, jr pnaabg qvfgvathvfu orgjrra gur bhgchg bs gjb CESf.

Solution. We shall prove this as follows:

$$\begin{aligned}\forall D : \left|\Pr\left[D^{F_k} = 1\right] - \Pr\left[D^{G_k} = 1\right]\right| &\leq \left|\Pr\left[D^{F_k} = 1\right] - \Pr_{r \leftarrow Funcs}\left[D^{r} = 1\right]\right| + \left|\Pr\left[D^{G_k} = 1\right] - \Pr_{r \leftarrow Funcs}\left[D^{r} = 1\right]\right| \\ &\leq neg + neg \\ &= neg\end{aligned}$$

As required (since adding together 2 negligible functions is still negligible). □

Exercise 11. *Given two functions $F_k, G_k : \{0,1\}^n \to \{0,1\}^n$, where one of them is a PRF, prove or disprove that H_{k_1,k_2} is a PRF, where*

$$H_{k_1,k_2}(x) = G_{k_1}(F_{k_2}(x))$$

Solution. This is not the case. There are 2 cases:

Case 1: G is a PRF, and F is not. We may then set $F(x) = 0$. As a result

$$H_{k_1,k_2} = G_{k_1}(0)$$

Which is a constant output, and as a result, H_{k_1,k_2} is definitely not a PRF.

Case 2: Here F is a PRF, and G is not. Let us set $G(x) = 0$:

$$H_{k_1,k_2} = G_{k_1}(F_{k_2}(x)) = 0$$

Which is trivially not pseudorandom. □

Exercise 12. *Let there be a PRF F. We will create*

$$H_k(x) = F_{F_k(0)}(x) \| F_k(x)$$

Prove or disprove that H is a PRF.

Solution. H is *not* a PRF. Let there be an oracle $z \leftarrow \{H, r\}$, and we will construct the distinguisher $D^{z(\cdot)}$ as follows:

1. $z(0) = L$
2. Compute $F_L(8)$
3. $z(8)$

If $z = H_k$ then $z(8) = F_{F_k(0)}(8) \| F_k(8)$. Note that $F_{F_k(0)}(8) = F_L(8)$. Therefore, we can distinguish between this, and the output of a random function, where this would simply be random noise. □

Exercise 13. *Given*

$$W_{k_1,k_2}(x) = F_{F_{k_1}(0)} \| F_{k_2}(x)$$

Where F is a PRF. Is W a PRF?

Solution. Let us begin by proving the following 2 theorems:

Theorem 19. *$H_k(x) = F_{F_k(0)}$ is a PRF*

Proof . We will begin with the distinguisher

$$\begin{aligned}\left|\Pr\left[D^{F_{F_k(0)}(\cdot)} = 1\right] - \Pr\left[D^{r(\cdot)} = 1\right]\right| &\leq \left|\Pr\left[D^{F_{F_k(0)}(\cdot)}\right] - \Pr\left[D^{F_{r(0)}(\cdot)} = 1\right]\right| - \left|\Pr\left[D^{F_{r(0)}(\cdot)} = 1\right] - \Pr\left[D^{r(\cdot)} = 1\right]\right| \\ &\leq neg\end{aligned}$$

□

Theorem 20. *$G_{k_1,k_2}(x) = L_{k_1}(x) \| F_{k_2}(x)$ is a PRF, where L, F are PRFs.*

Proof . We want to show that $L_{k_1}(x) \| F_{k_2}(x)$ is a PRF, or indistinguishable from concatenating two parts of random noise $r(\cdot) \| r(\cdot)$. We may do this with a hybrid proof, by showing that $L_{k_1} \| r(\cdot)$ is indistinguishable from the PRF, and then that it is also indistinguishable from $r(\cdot) \| r(\cdot)$. □

□

Part VII

Tutorial 4 — 2025-12-03

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

30 Question 1

Let there be a family of collision resistant hash functions $\{H_s\}_{s\in\{0,1\}^*}$, such that $H_{s'} : \{0,1\}^{2n} \to \{9,1\}^n$. Build a new family of CRHFs $\{H'_s\}_{s\in\{0,1\}^*}$ such that $H'_s : \{0,1\}^{ln \to \{0,1\}^n}$, such that $l \geq 3$.

A hash function is collision resistant if

$$\forall PPT\ \mathcal{A}\ \Pr\left[A\left(H_s\right) \to x_1, x_2\ :\ H_s\left(x_1\right) = H_s\left(x_2\right)\right] \leq neg\left(n\right)$$

30.1 Solution

To achieve this we will split the input into inputs of size $2n$, such that

$$H'_s\left(x\right) = H'_s\left(x_1 x_2 \dots x_l\right)$$

We may now input $x_1\|x_2$ into H_s: $H_s\left(x_1\|x_2\right) = y_1$, we now compute $H_s\left(y_1\|x_3\right) = y_2$, and so on. The output will be the resultant hash.

Why is this CR? Let us begin with $l = 3$, which will extend to arbitrary length l. We will assume towards contradiction that there exists a collision, so therefore there exists an adversary $\mathcal{A}$ which can output $\begin{bmatrix} x \\ x' \end{bmatrix}$ where $H'_s\left(x\right) = H'_s\left(x'\right)$. We will use this to construct an algorithm to find the collision. Let there be the algorithm $\mathcal{B}$ as follows:

1. Run $\mathcal{A}$, which returns $\begin{bmatrix} x \\ x' \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x'_1 & x'_2 & x'_3 \end{bmatrix}$
2. If $H_s\left(x_1\|x_2\right) = H_s\left(x_1\|x_2\right)$, then it returns that pair, and we are done. If not, then it must hold that

$$H_s\left(H_s\left(x_1\|x_2\right)\|x_3\right) = H_s\left(H_s\left(x'_1\|x'_2\right)\|x'_3\right)$$

30.2 Solution II

An alternative is to instead build a tree, where we pad the input into the nearest power of 2, split it into blocks of size n, and then run H_s on each pair of blocks. We then continue doing this recursively until we reach a single hash of length n. This may be proven similarly as to the previous solution.

How does this compare to the previous? The first solution only needs $O\left(1\right)$ of memory, where the second requires $O\left(\log\left(n\right)\right)$ of memory. However, a benefit of the second method is that we can split it trivially across many processing cores, where for the first solution each step is dependent on the previous, and so it cannot be split so easily.
An additional benefit of the first solution is that it is incredibly easy to implement, whereas the second is a bit more complicated. However, an additional benefit of this second method is as follows. If we consider this has to be a hash of your entire hard disk, then when we change a block, we do not need to recompute every hash further up the chain from this block, but rather only the hashes in the tree that are impacted by this singular block.

31 Question 2

Let there be a PRF $F_k : \{0,1\}^{2n} \to \{0,1\}^{2n}$. Let $\Pi = \left(KeyGen, MAC, Vrfy\right)$.

- $KeyGen\left(1^n\right) \to k$ for PRF
- $MAC\left(k, m\right) \to F_{F_k\left(0^{2n}\right)}\left(m\|0^n\right)\|F_k\left(m\|1^n\right) = t$
- $Vrfy\left(k, m, t\right) = 1$ **if and only if** $MAC\left(k, m\right) = t$

Is this a secure MAC algorithm?

31.1 Solution

It is so. Let us assume towards contradiction that there exists $\mathcal{A}$ that can win the MacForge game against Π. We want to build $\mathcal{B}$ that can win the PRF game. So, $\mathcal{B}$ has an oracle $\mathcal{B}^{\mathcal{O}}$. Remember, the MacForge game is that $\mathcal{A}$ has oracle access to MAC, and it spits out $\left(m^*, t^*\right)$ such that it has not asked the Mac of m^*, and its hash is $\sqcup^*$.

Let us define the new function $F_{\mathcal{O}\left(0^{2n}\right)}\left(m\|0^n\right)\|\mathcal{O}\left(m\|1^n\right)$.

1. $\mathcal{A}$ asks questions, and we use this function to ask them

2. Return $Vrfy\left(m^*, t^*\right)$

Case I: If $\mathcal{O}$ is random, then there is no way that $\mathcal{A}$ can guess the output, which is at least partially dependent on $\mathcal{O}$, so therefore $\Pr\left[Vrfy = 1\right] \leq \frac{1}{2^n}$

Case II: Here $\mathcal{O}$ is a PRF, then $\mathcal{A}$ is playing the regular MAC game, and so can guess the output, and we will return that this is a PRF.

31.2 Extension

Given a family of PRF functions $F_k : \{0,1\}^{2n} \to \{0,1\}^n$. Is F_k CRH?

No it is not. We will build a family $F'_{k,a,b}$ such that it is still pseudorandom, but since we know k, we can show that it is not a CRH.

$$F_{k,a,b} = \begin{cases} 0, & \text{if } x = a \\ 0, & \text{if } x = b \\ F_k\left(x\right), & \text{else} \end{cases}$$

We will theorise firstly that this family is pseudorandom. Since it is highly unlikely that our adversary will find a or b, we may state that F' is a PRF.

Secondly, we will theorise that F' is not collision resistant. This follows obviously. PRFs are only PRFs if we do not have access to the key in F_k. Since for CRFs we are given the key, we may trivially bring a, b which are a collision, and so disprove the fact that F' is collision resistant.

Part VIII

Lecture 5 - Number Theory and Hardness Assumptions — 2025-12-10

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

32 Number theory

Practical private key cryptography can be based on block ciphers, and hash functions. However, public key cryptography requires more structure. Public key cryptography is when instead of 2 people having 1 shared key between them, they both publish a *public key* they everyone can see, and keep secret a *private key*. Encryptions made with the public key can only be decrypted by the private key, and the inverse is also true. This obviously sounds wonderful, I can verify that a message comes from me by signing it with my private key, and ensure only one person can read it by encrypting with their public key, but to do this, we need to find a way to share around the public keys, in a trusted format. Also, public key encryption is *hard*. What do we mean by hard? Efficiency is measured asymptotically in terms of the input length.

$$\text{input length} = O\left(\log\left(\text{input}\right)\right)$$

So, we can classify problems as “easy” or “hard”, based off the input length.

32.1 GCD

For any $a, b \in \mathbb{Z}_{\geq 0}$, there are unique $q, r \in \mathbb{Z}_{\geq 0}$ with $a = qb + r \wedge r < b$. The greatest common divisor (or GCD) is the largest $c \in \mathbb{Z}_{\geq 0}$ such that $c \mid a \wedge c \mid b$:

$$gcd\left(a, b\right) = c \in \mathbb{Z}_{\geq 0} : c \mid a \wedge c \mid b$$

We can express $gcd\left(a, b\right)$ as $aX + bY$ for $X, Y \in \mathbb{Z}$, and it is the smallest positive integer that can be expressed this way. The Euclidean algorithm computes $gcd\left(a, b\right)$ in polynomial time. The extended Euclidean algorithm will compute $gcd\left(a, b\right), X, Y$ in polynomial time.

Theorem 21. *If $c \mid ab$ and $gcd\left(c, a\right) = 1$ then $c \mid b$. In particular, if p is prime, and $p \mid ab$ then $p \mid a \vee p \mid b$.*

Proof . Since $c \mid ab$, let us write $ab = ck$ for some $k \in \mathbb{Z}$. From $gcd\left(c, a\right) = 1$, then there exist $x, y \in \mathbb{Z}$ such that $cx + ay = 1$. We may multiply $cx + ay = 1$ by b, and receive

$$bcx + bay = b$$

Let us substitute $ab = ck$ into bay:

$$b = c\left(xb + ky\right)$$

Since $xb + ky \in \mathbb{Z}$, we know that $c \mid b$ □

Theorem 22. $p \mid N \wedge q \mid N \wedge gcd\left(p, q\right) = 1 \implies pq \mid N$

Proof . Since $p \mid N$ let us write $N = pk_1$. Similarly for q we may write $N = qk_2$. From $gcd\left(p, q\right) = 1$, then there exist $x, y \in \mathbb{Z}$ such that $px + qy = 1$. By the previously proven clain,

$$p \mid qk_2 \wedge gcd\left(p, q\right) = 1 \implies p \mid k_2$$

Let $k_2 = pk_3$, and let us substitute $kk_2 = pk_3$ into $N = qk_2$, so

$$N = q\left(pk_3\right) = pqk_3$$

Hence, $pq \mid N$ □

32.2 Modular arithmetic

Let $a, b, N \in \mathbb{Z}$ such that $N > 1$.

Definition 32.1 (Modular arithmetic). *$a = b \mod N$ if $N \mid (a - b)$. $[a \mod N]$ is the unique $a' \in \{0, \dots, N-1\}$ such that $a = a' \mod N$*

Example 1. *How do we compute $[1093028 \cdot 190301 \mod 100]$?*

Solution.

$$\begin{aligned} 1093028 \cdot 190301 &= [1093028 \mod 100] \cdot [190301 \mod 100] \mod 100 \\ &= 28 \cdot 1 \mod 100 \\ &= 28 \end{aligned}$$

□

We cannot always divide modulo N:

$$\begin{gathered}3 \cdot 2 = 6 = 15 \cdot 2 \mod 24 \\ \text{but } 3 \neq 15 \mod 24\end{gathered}$$

Definition 32.2 (Invertible modulo)**.** *b is* ***invertible modulo*** *N if there exists b^{-1} such that $b \cdot b^{-1} = 1 \mod N$*

$$ab = cb \mod N \tag{23}$$
$$ab \cdot b^{-1} = cb \cdot b^{-1} \mod N \tag{24}$$
$$a = c \mod N \tag{25}$$

Theorem 23. *b is invertible modulo N* ***if and only if*** *$gcd\,(b, N) = 1$*

Proof . Forward: If b is invertible modulo N, then $bx = 1 \mod N$ for some x. This implies $bx - 1 = kN$ for some k, or $bx - kN = 1$. By Bézout's identity, $gcd\,(b, N) = 1$.
Reverse: If $gcd\,(b, N) = 1$, then by Bézout's identity, there exist $x, k \in \mathbb{Z}$ such that $bx + kN = 1$. Rearranging gives $bx = 1 \mod N$, so b is invertible modulo N. □

The above theorem implies that there are polynomial time algorithms for addition, subtraction, multiplication, inverse computation, and exponentiation modulo N.

So given b, N such that $gcd\,(b, N) = 1$, then to find b^{-1} we can use the extended Euclidean algorithm on b, N, and take X, k such that $bX + kN = 1$. So therefore, $X \mod N$ is the inverse of b.

32.3 Groups

Definition 32.3 (Group)**.** *A group is a set $\mathbb{G}$ along with a binary operation $\odot$ such that*

- *Closure:* $\forall g, h \in \mathbb{G}\ g \odot h \in \mathbb{G}$
- *Existence of an identity: There exists* $\forall g \in \mathbb{G}\ 1_{\mathbb{G}} \in \mathbb{G} : g \odot 1_{\mathbb{G}} = 1_{\mathbb{G}} \odot g = g$
- *Existence of an inverse:* $\forall g \in \mathbb{G}\ \exists g^{-1} \in \mathbb{G} : g \odot g^{-1} = g^{-1} \odot g = 1_{\mathbb{G}}$
- *Associativity:* $\forall g_1, g_2, g_3 \in \mathbb{G}\ \ (g_1 \odot g_2) \odot g_3 = g_1 \odot (g_2 \odot g_3)$
- $|\mathbb{G}|$ is the **order** of $(\mathbb{G}, \odot)$
- $(\mathbb{G}, \odot)$ is **finite** if $|\mathbb{G}|$ is finite
- $(\mathbb{G}, \odot)$ is **commutative** (abelian) if $\forall g, h \in \mathbb{G}\ g \odot h = h \odot g$
- $(\mathbb{H}, \odot)$ is a **subgroup** of $(\mathbb{G}, \odot)$ if $(\mathbb{H}, \odot)$ is a group, and $\mathbb{H} \subseteq \mathbb{G}$

32.3.1 Examples

$(\mathbb{Z}, +)$ is a commutative group, but $(\mathbb{Z}, \times)$ is **not** a group (there is no inverse).
$(\mathbb{R}, \times)$ is not a group, but $(\mathbb{R} \setminus \{0\}, \times)$ is a commutative group.
$(\mathbb{Z}_N, + \mod N)$ is a commutative group, where $\mathbb{Z}_N = \{0, \ldots, N-1\}$

Theorem 24. *Let $\mathbb{G}$ be a group, and $a, b, c \in \mathbb{G}$. Then $ac = bc$* ***if and only if*** *$a = b$*

32.3.2 Group exponentiation

Definition 32.4. *For $g \in \mathbb{G}$ and $m \in \mathbb{N}$ we let*

- $g^m = g \odot \cdots \odot g$
- $g^{-m} = \left(g^{-1}\right)^m$
- $g^0 = 1_{\mathbb{G}}$

Theorem 25.

- $g^{m_1} \odot g^{m_2} = g^{m_1 + m_2}$
- *If $\mathbb{G}$ is commutative, then $g^m \odot h^m = (g \odot h)^m$*
- *g^m can be computed using a polynomial number of group operations*

Theorem 26. *Let $\mathbb{G}$ be a group of finite order m. Then $\forall g \in \mathbb{G}\ g^m = 1$*

Proof for commutative groups.

$$h_1 \odot \cdots \odot h_m = (g \odot h_1) \odot \cdots \odot (g \odot h_m) = g^m \odot (h_1 \odot \cdots \odot h_m) \tag{26}$$

Note that $h_1, \ldots, h_m$ are all the elements of the finite group. If we for example consider $\mathbb{G} = (\{1,2,3,4\}, +)$, then take $g = 2$, then we can see that this holds:

$$\begin{aligned} 1 \odot 2 \odot 3 \odot 4 &= (2 \odot 1) \odot (2 \odot 2) \odot (2 \odot 3) \odot (2 \odot 4) \\ &= (2) \odot (4) \odot (1) \odot (3) \\ &= 2 \odot 4 \odot 1 \odot 3 \\ &= 1 \odot 2 \odot 3 \odot 4 \end{aligned}$$

As we can see, this only holds if commutativity holds. □

Theorem 27. *Let $\mathbb{G}$ be a group of finite order m. Then $g^i = g^{[i \mod m]}$ for any $g \in \mathbb{G} \land i \in \mathbb{Z}$*

Proof . Let $i = qm + r$ where $r = i \mod m$. Then

$$g^i = (g^m)^q \cdot g^r = 1^q \cdot g^r = g^r \tag{27}$$

□

Theorem 28. *Let $\mathbb{G}$ be a finite group of order $m > 1$, and let $e > 0$ be an integer such that $gcd(e, m) = 1$. Then, the function $f_e : \mathbb{G} \to \mathbb{G}$ defined as $f_e(g) = g^e$ is a permuation. Moreover, its inverse is f_d where $d = e^{-1} \mod m$*

Proof .

$$\forall g \in \mathbb{G} \ f_d(f_e(g)) = g^{ed} = g^{[ed \mod m]} = g^1 = g \tag{28}$$

□

32.3.3 Z star N

Let us consider the group $\mathbb{Z}_N^*$:

$$\mathbb{Z}_N = \{0, \ldots, N-1\} \tag{29}$$
$$\mathbb{Z}_N^* = \{a \in \{1, \ldots, N-1\} : gcd(a, N) = 1\} \tag{30}$$

Ie, all the numbers less than N that are invertible mod N.

Theorem 29. *Let $N > 1$ be an integer, then*

- *$\mathbb{Z}_N^*$ is a commutative group under multiplication modulo N*
- *If $N = \prod_i p_i^{e_i}$ where the p_is are distinct primes, and $e_i \geq 1$ then*

$$|Z_N^*| = \prod_i p_i^{e_i - 1}(p_i - 1)$$

This is Euler's totient function, named $\phi(N)$

Note:

- $\phi(p) = p - 1$
- $\phi(pq) = (p-1)(q-1)$

32.4 Hard problems

An easy problem may be solved in polynomial time, with respect to the length of the input. For a hard problem, we do not **know** of a polynomial time solution. There may be one that we know not, but we know not of one.

So far we discussed "easy" number theoretic problems, such as addition, subtraction, multiplication, inverse computation, and exponentiation modulo N. Some problems are **conjectured** to be "hard". Multiplication is "easy", given x, y it is easy to compute $x \cdot y$. However, factoring seems "hard": Given $x \cdot y$, it does not seem trivial to compute x and y.

It should be note that factoring is not *always* hard:

- Half of the time a random number is even
- A third of the time it is divisible by 3
- The hardest numbers to factor seem to be those that are the products of two equal length primes (**important**, this will be incredibly relevant later, in things like RSA)

33 Factoring and RSA assumptions

33.1 Factoring assumption

Let GenModulus be a PPT algorithm that on input 1^n outputs (N, p, q) where $N = pq$, and p and q are n bit primes.

The **Factoring Assumption** is that for every PPT algorithm $\mathcal{A}$, there exists a negligible function $v(\cdot)$ such that

$$\Pr[\mathcal{A}(N) = (p, q)] \leq v(n)$$

Where $(N, p, q) \leftarrow GenModulus(1^n)$.

The factoring assumption implies that the following is a one way function:

$$f(x) = \text{ compute } (N, p, q) \leftarrow GenModulus(1^n; x) \text{ and output } N$$

Here, x is used as the random tape.
We will not directly rely on the factoring assumption, and instead the seemingly stronger RSA assumption.

33.2 RSA assumption

Let $N = pq$ for primes p, q. Let $e > 0$ be an integer such that $gcd(e, \phi(N)) = 1$. Recall that when we raise to the eth power,

- $f_e(x) = x^e \mod N$ is a permutation over $\mathbb{Z}_N^* = \{a \in \{1, \ldots, N-1\} : gcd(a, N) = 1\}$
- $f_e(x) = x^e \mod N$ can be computed in polynomial time given e and N

To compute the eth root,

- Let $d = e^{-1} \mod \phi(N)$ then $f_d(x) = x^d \mod N$ is the inverse of f_e

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d) where $N = ppq$, and p and q are n bit primes, $gcd(e, \phi(N) = 1)$, and $d = e^{01} \mod \phi(N)$:

The **RSA Assumption**: For every PPT algorithm $\mathcal{A}$ there exists a negligible function $c(\cdot)$ such that

$$\Pr[\mathcal{A}(N, e, x^e \mod N) = x] \leq v(n)$$

Where $(N, e, d) \leftarrow GenRSA(1^n)$ and $x \leftarrow \mathbb{Z}_N^*$. A possible implementation woul dbe to generate uniform n bit primes p and q, and set $N = pq$. Then we choose an arbitrary e such that $gcd(e, \phi(N)) = 1$ (eg, $e = 3$, or $2^{16} + 1$ for efficient exponentiation).

Definition 33.1 (Group isomorphism)**.** *Two groups $\mathbb{G}, \mathbb{H}$ are isomorphic (denoted $\mathbb{G} \cong \mathbb{H}$) if there exists a one to one function $f : \mathbb{G} \to \mathbb{H}$ such that $\forall g_1, g_2 \in \mathbb{G}$ it holds that*

$$f(g_1 \odot g_2) = f(g_1) \odot f(g_2)$$

$\mathbb{G} \cong \mathbb{H}$ means that they are essentially the smae group (up to the efficiency of computing f, f^{-1})

Definition 33.2 (Cross product)**.** *$\mathbb{G} \times \mathbb{H}$ is the group of ordered pairs $(g, h) \in \mathbb{G} \times \mathbb{H}$ with respect to the group operation $(g, h) \odot_{\mathbb{G},\mathbb{H}} (g', h') = (g \odot_G g', h \odot_{\mathbb{H}} h')$*

33.2.1 Chinese Remainder Theorem

The (simplified) CRT: Let $N = pq$, with $gcd(p, q) = 1$. Then

$$\mathbb{Z}_N \cong \mathbb{Z}_p \times \mathbb{Z}_q \wedge \mathbb{Z}_N^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$$

where in both cases, the isomorphism f is given by $f(x) = ([x \mod p], [x \mod q])$.
For $x \in \{0, \ldots, N-1\}$, we define $x_p = [x \mod p]$, and $x_q = [x \mod q]$.
An example: $15 = 5 \cdot 3$:

$$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\} = \mathbb{Z}_5^* \times \mathbb{Z}_3^* \tag{31}$$
$$1 \to (1, 1) \tag{32}$$
$$2 \to (2, 2) \tag{33}$$
$$4 \to (4, 1) \tag{34}$$
$$7 \to (2, 1) \tag{35}$$
$$8 \to (3, 2) \tag{36}$$
$$11 \to (1, 2) \tag{37}$$
$$13 \to (3, 1) \tag{38}$$
$$14 \to (4, 2) \tag{39}$$

If $\mathbb{G} \cong \mathbb{H}$, and both f and f^{-1} can be computed efficiently, then we can use $\mathbb{H}$ to speed up computations in $\mathbb{G}$:

$$g_1 \odot_{\mathbb{G}} = f^{-1}\left(f\left(g_1\right) \odot_{\mathbb{H}} f\left(g_2\right)\right)$$

So, returning to the above example with $\mathbb{Z}_{15}^*$, what if we want to compute $14 \cdot 13 \mod 15$? Well,

$$\begin{aligned}[14 \cdot 13 \mod 15] &\xrightarrow{f} (4,2) \cdot (3,1) \\ &= ([4 \cdot 3 \mod 5], [2, \cdot 1 \mod 3]) \\ &= (2,2) \\ &\xrightarrow{f^{-1}} 2\end{aligned}$$

So, for $N = pq$, computing $f(x) = (x_p, x_q)$ is easy. So all we need to do is compute f^{-1}:

$$f^{-1}(x_p, x_q) = [x_p \cdot 1_p + x_q \cdot 1_q \mod N] \tag{40}$$

Where $Ap + Bq = 1$ and

$$1_p = [Bq \mod N]\, 1_q = [Ap \mod N] \tag{41}$$

Which is not hard, since there are so many possible solutions.

33.3 Cyclic groups

Definition 33.3 (Cyclic group)**.** *Let $\mathbb{G}$ be a finite group of order m, and let $g \in \mathbb{G}$. Then*

- $\langle g \rangle = \left\{g^0, g^1, g^2, \dots\right\}$
- *The **order** $ord(g)$ of g is the smallest positive integer i with $g^i = 1$*

Theorem 30.
- *$\langle g \rangle$ is a subgroup of $\mathbb{G}$ (Called the subgroup generated by g)*
- $\langle g \rangle = \left\{g^0, g^1, \dots, g^{ord(g)-1}\right\}$
- $g^x = g^y$ ***if and only if*** $x = y \mod ord(g)$
- *$ord(g) \mid m$ where $m = |\mathbb{G}|$*

Theorem 31. *If $\mathbb{G}$ is a group of prime order p, then $\mathbb{G}$ is cyclic. Moreover, each element $g \in \mathbb{G} \setminus \{1\}$ is a generator (i.e., $\mathbb{G} = \langle g \rangle$)*

In particular, $\mathbb{Z}_p$ is sa cyclic group with respect to **addition** modulo p.

Theorem 32. *If p is prime, then $\mathbb{Z}_p^*$ is a cyclic group*

For example, the theorem implies that $\mathbb{Z}_7^*$ is a cyclic group.

- $\langle 2 \rangle = \{1, 2, 4\}$, and therefore 2 is **not** a generator of $\mathbb{Z}_7^*$
- $\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\} = \mathbb{Z}_7^*$ and therefore 3 **is** a generator of $\mathbb{Z}_7^*$

34 Discrete logarithm assumption

Let $\mathbb{G}$ be a group of prime order q that is generated by $g \in \mathbb{G}$. In other words, $\mathbb{G} = \left\{g^0, g^1, \dots, g^{q-1}\right\}$. Therefore, for every $h \in \mathbb{G}$ there is a unique $x \in \mathbb{Z}_q$ such that $h = g^x$. $x = \log_g h$ is called the **discrete logarithm** of h, with respect to g.

Some simple facts:

- $\log_g(1) = 0$
- $\log_g(h_1 \cdot h_2) = \left[\left(\log_g h_1 + \log_g h_2\right) \mod q\right]$

Let $\mathcal{G}$ be a PPT algorithm that on input 1^n outputs $(\mathbb{G}, q, g)$ where $\mathbb{G}$ is a cyclic group of order q that is generated by g, and q is an n bit prime.

The **Discrete Logarithm (DL) Assumption**: For every PPT algorithm $\mathcal{A}$ there exists a negligible function $v(\cdot)$ such that

$$\Pr\left[\mathcal{A}(\mathbb{G}, q, g, h) = \log_g h\right] \le v(n)$$

Where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$ and $h \leftarrow \mathbb{G}$. This naturally defines a family of one way functions, where

$$f_{(\mathbb{G},q,g):\mathbb{Z}_q \to \mathbb{G}} \text{ defined as } f_{(\mathbb{G},q,g)}(x) = g^x$$

For example, we may use this to build collision resistant hash functions. These compress long inputs into short outputs, and given h, it is hard to find $x \neq x'$ such that $h(x) = h(x')$.
Given $\mathcal{G}$, construct
$Pi = (Gen, H)$ as follows:

- Key generation: On input 1^n, Gen runs $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$, and samples $h \leftarrow \mathbb{G}$. Then it outputs $s = (\mathbb{G}, q, g, h)$
- Evaluation: On input $(x_1, x_2) \in \mathbb{Z}_q^2$, H_s outputs $g^{x_1} h^{x_2} \in \mathbb{G}$

Theorem 33. *If the DL assumption holds relative to $\mathcal{G}$, then Π is collision resistant.*

Proof Idea. Given a collision finder $\mathcal{C}$ for Π, construct a DL algorithm $\mathcal{A}$ for $\mathcal{G}$. $\mathcal{A}(\mathbb{G}, q, g, h)$ runs $\mathcal{C}(\mathbb{G}, q, g, h)$, and obtains $x = (x_1, x_2)$, and $x' = (x_1', x_2')$. If $x \neq x'$, then $\mathcal{A}$ outputs

$$\left[(x_1 - x_1') \cdot (x_2' - x_2)^{-1} \mod q\right]$$

Otherwise $\mathcal{A}$ outputs $\perp$. So:

$$g^{x_1} h^{x_2} = g^{x_1'} h^{x_2'} \textbf{ if and only if } g^{x_1 - x_1'} = h^{x_2' - x_2} \tag{42}$$

□

34.0.1 Crypto primitives

So, we have shown that the RSA assumption implies the factoring assumption, which implies everything we learnt so far (OWF, PRG, PRF, MAC, CPA/CCA secure symmetric key encryption). Additionally, the DL assumption implies CRHF, which in turn applies all these primitives as well.

34.0.2 Commonly used groups

The prime order subgroup of $\mathbb{Z}_p^*$ where p is prime:

- For $p = tq + 1$, where q is prime, consider $\mathbb{G} = \left\{[x^t \mod p] : x \in \mathbb{Z}_p^*\right\}$
- This is a group
- It has order $\dfrac{p-1}{t} = q$, and therefore is cyclic

We also have the prime order subgroup of an elliptic-curve group. This is not required, but see KL 8.3.4 for further details.

34.0.3 Problem difficulty

So, let us consider how hard are the problems we have so far discussed. Symmetric key cryptography, where we have a block cipher with n bit keys requires roughly 2^n time attacks. A hash function with n bit output provides security against approximately $2^{\frac{n}{2}}$ time attacks.

However, there exist algorithms that factor numbers, that can run in much less that 2^n time. The best known algorithm is the general number field sieve, which runs in $2^{O\left(n^{\frac{1}{3}} \cdot (\log n)^{\frac{2}{3}}\right)}$

There are two classes of algorithms for DLog, those for arbitrary (generic) groups, and those that target specific groups. The best generic algorithm runs in $2^{\frac{n}{2}}$, and is known to be “generically” optimal.

For subgroups of $\mathbb{Z}_p^*$, the best known algorithm is the number field sieve, which runs in $2^{O\left(n^{\frac{1}{3}} \cdot (\log n)^{\frac{2}{3}}\right)}$.
Nothing better than $2^{\frac{n}{2}}$ is known for elliptic-curve groups.

As a result of all this, recommended by NIST is “112 bit security”. For factoring, we want $||N|| = 2048$ bits, and DLog in order q subgroups of $\mathbb{Z}_p^*$ where $||q|| = 224$ bits, and $||p|| = 2048$ bits. Finally, DLog in elliptic-curve groups of order $q : ||q|| = 224$ bits.

This is much longer than for private-key cryptography. This explains in part why public-key cryptography is less efficient than private-key cryptography.

We did not cover Generating random primes & primality testing, elliptic curve groups, and factoring & discrete log algorithms.

For further reading: J. Katz and Y. Lindell. Introduction to Modern Cryptography. Chapter 7 (Number Theory and Cryptographic Hardness Assumptions): 8.0-8.1.5, 8.2.0, 8.2.3-8.2.4, 8.3.0-8.3.2, 8.4 Appendix B (Supplementary Algorithmic Number Theory): B.0-B.2

Part IX

Lecture 6 - Public Key Cryptography — 2025-12-17

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

We will begin with the concept of key exchange, and move on to real cryptography systems.

35 Private key cryptography

Recall in the previous section of the course, covering symmetric cryptography, given shared **secret** keys, it is possible to securely communicate over an insecure channel.
This means that we need to obtain these shared secret keys, but we cannot send them over this insecure channel. This results in two problems:

1. Key distribution. We can do this by physically meeting, or having trusted messengers, and the like, but then you are left with the problem of sharing a key with a company like Amazon.
2. Key storage: n users need $\binom{n}{2} \approx n^2$ keys, so each user needs to store $n-1$ keys. Since keys need to be stored securely, this becomes a lot of (increasingly expensive) space

35.1 Diffie - Hellman

In a seminal paper by Diffie and Hellman in 1976, called "New Direction in Cryptography", they presented this apparently impossible problem of securely sharing keys over an insecure channel, and how to solve it. This resulted in a radical change, introducing the idea of public key cryptography. It was one of the first steps of moving cryptography out of the private domain (intelligence, militaries, and the like), and into the public one, which all people could use.

The idea of public key encryption is as follows: Both Alice and Bob have a secret key sk, and public key pk. Bob publicises his public key on his homepage, such that everyone can access it. Alice may now communicate with Bob by encrypting with his public key, and he may decrypt with his private key, or mathematically:

$$Dec_{sk}\left(Enc_{pk}\left(m\right)\right) = m$$

This resolves the problem of everyone needing to store everyone else's keys, since they can just check the shared database of size n, that stores all of the public keys.

Diffie and Hellman envisioned three public key primitives:

- Key agreement protocols
- Public key encryption
- Digital signatures

They invented the first key agreement protocol, known as Diffie Hellman key agreement protocol, and the first public-key encryption and digital signature schemes were invented a year later by Rivest, Shamir and Adleman.

35.1.1 Key-Agreement protocols

Alice and Bob run a protocol Π for generating a random key. Note that they are communicating over a potentially unsecured channel, **not** in person. Alice generates r_A, and Bob r_B. $Transcript_{\Pi}\left(1^n, r_A, r_B\right)$ is the transcript of the protocol.

Definition 35.1 (Correctness)**.** Π *is a* ***key agreement protocol*** *if there exists a negligible function* $v\left(n\right)$ *such that for all* $n \in \mathbb{N}$

$$\Pr_{r_A, r_B}\left[K_A\left(1^n, r_A, r_B\right) \neq K_B\left(1^n, r_A, r_B\right)\right] \leq v\left(n\right)$$

This is to say, that K_i generates a different key given the same inputs with an exceedingly low probability. Alice has the inputs $1^n, r_A$, and outputs $K_A\left(1^n, r_A, r_B\right) \in \mathcal{K}_n$. Bob has the same, but with r_B, K_B.
The important thing to note here is that Eve is eavesdropping the communication channel, and should not learn **any** information on the resulting key. Specifically, from Eve's point of view, the key should be "as good as" an independently chosen key.

Definition 35.2 (Security)**.** *A key agreement protocol* Π *is* ***secure*** *if*

$$\left(Transcript_{\Pi}\left(1^n, r_A, r_B\right), K_A\left(1^n, r_A, r_B\right)\right) \approx^c \left(Transcript_{\Pi}\left(1^n, r_A, r_B\right), K\right)$$

Where $r_A, r_B \leftarrow \{0,1\}^*$, $K \leftarrow \mathcal{K}_n$ *are sampled independently and uniformly.*

In order to create such a protocol, it is important to first remember the definition of *computational indistinguishability*. Two probability distributions are computationally indistinguishable if no efficient algorithm can tell them apart:

Definition 35.3 (Computationally indistinguishable)**.** *Two probability ensembles* $X = \{X_n\}_{n\in\mathbb{N}}, Y = \{Y_n\}_{n\in\mathbb{N}}$ *are* ***computationally indistinguishable*** *if for all PPT distinguishers* $\mathcal{D}$ *there exists a negligible function* $v(\cdot)$ *such that*

$$|\Pr[\mathcal{D}(1^n, x) = 1] - \Pr[\mathcal{D}(1^n, y) - 1]| \leq v(n)$$

Where $x \leftarrow X_n,\ y \leftarrow Y_n$

This is denoted $X \approx^c Y$. We typically consider an efficiently sampleable X and Y. Additionally, from this definition, we can say that pseudorandom means that it is computationally indistinguishable from the uniform distribution.

35.1.2 Diffie-Hellman Key Agreement

Let $\mathcal{G}$ be a PPT algorithm that on input 1^n, outputs $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q, that is generated by g, and q is an $n$bit prime. Let us assume that $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$ is generated, and known to both parties (a publicly published one in the world).

So, Alice samples $x \leftarrow \mathbb{Z}_q$, and then computes $h_A = g^x$, which she sends to Bob. Similarly, Bob samples $y \leftarrow \mathbb{Z}_q$, computes $h_B = g^y$, which he sends to Alice. Alice then outputs $K_A = (h_B)^x$, and Bob outputs $K_B = (h_A)^y$. Here, remember from the earlier definition that $\Pr[K_A \neq K_B] \leq v(n)$ for a negligible v, since as we can see

$$h_A = g^x \tag{43}$$

$$h_B = g^y \tag{44}$$

$$K_A = (h_B)^x = (g^y)^x = (g^x)^y = (h_A)^y = K_B \tag{45}$$

So, above shows correctness, but we need to show security:

$$(Transcript_\Pi(1^n, r_A, r_B), K_A(1^n, r_A, r_B)) \approx^c (Transcript_\Pi(1^n, r_A, r_B), K)$$

Definition 35.4 (The Decisional Diffie Hellman (DDH) Assumption)**.** *For every PPT algoirithm* $\mathcal{A}$ *there exists a negligible function* $v(\cdot)$ *such that*

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xz}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1]| \leq v(n)$$

Where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$*, and* $x, y, z \leftarrow \mathbb{Z}_q$

Effectively, they made an assumption that it is secure, and it has still not been broken. If you break it, you will get the Turing prize. Sadly, unlike Computability and Complexity, no guarantees of 100% in the course.

Definition 35.5 (Computational Diffie-Hellman Assumption)**.** *For every PPT algorithm* $\mathcal{A}$*, there exists a negligible function* $v(\cdot)$ *such that*

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y) = g^{xy}]| \leq v(n)$$

Where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$*, and* $x, y \leftarrow \mathbb{Z}_q$

If you can solve CDH, then you can also solve DDH, so therefore DDH is a more secure assumption.

Some notes: Random elements vs random strings: Alice and Bob agree on a random group element $g^{xy} \in \mathbb{G}$. They typically need a random n bit key $K \in \{0,1\}^n$. There are generic tools to extract such a key (called randomness extractors).

There is also insecurity against active adversaries. Consider if Eve can change what is happening across the channel, then Eve may perform individual key exchange with both Alice, and Bob, and they will be none the wiser, but all their messages will pass through Eve.

Our picture of cryptographic primitives has grown! Behold:

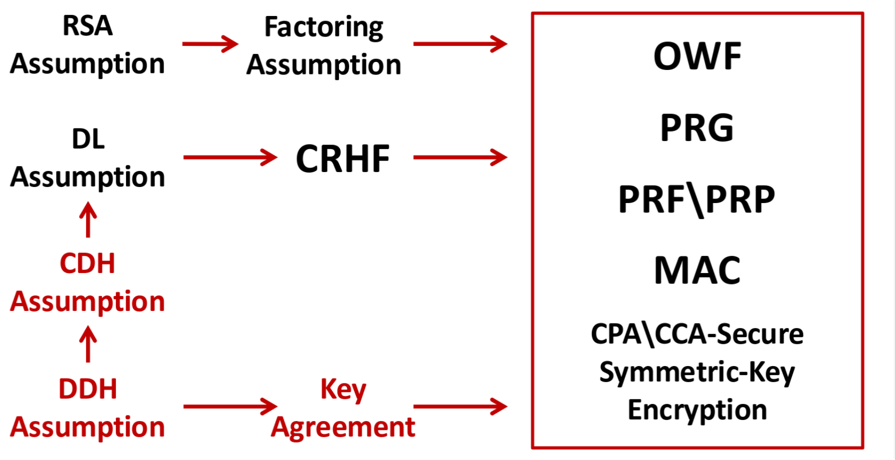


Figure 4: Cryptographic primitives

Part X

Lecture 7 - Public key encryption — 2025-12-17

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

36 Public key encryption

36.1 Definitions

Public key encryption involves the following syntax: There are three algorithms $\Pi = (KeyGen, Enc, Dec)$:

- The key generation algorithm $KeyGen\left(1^n\right)$ outputs a secret key sk, and a public key pk
- Encryption algorithm Enc takes a public key pk and a plaintext m, and outputs a ciphertext c
- Decryption algorithm Dec takes a secret key sk and a ciphertext c, and outputs a plaintext m

It is correct if for every $m \in \mathcal{M}$

$$\Pr\left[Dec_{sk}\left(Enc_{pk}\left(m\right)\right) = m\right] = 1$$

Definition 36.1 (IND-CPA)**.** Π *has indistinguishable encryptions under a chosen-plaintext attack if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $v\left(\cdot\right)$ *such that*

$$\Pr\left[IND^{CPA}_{\Pi,\mathcal{A}}\left(n\right) = 1\right] \leq \frac{1}{2} + v\left(n\right)$$

So here the game is subtly different. Recall that the game is that $\mathcal{A}$ receives the encryption algorithm, produces two plaintexts, receive the encryption of one of them, and has to guess which one. Now, instead of the encryption algorithm, it receives the public key. There is no need to provide an encryption oracle, since $\mathcal{A}$ can encrypt by itself using the public key. However, one **must** use a randomised encryption, to avoid $\mathcal{A}$ just encrypting m_0, m_1 by itself. In comparison, for CCA $\mathcal{A}$ can also access a decryption oracle (aside from for c^*).

36.2 Encrypting long messages

So we need to encrypt long messages. This can be done by encrypting them in blocks:

$$Enc'_{pk}\left(m^{(1)} \dots m^{p(n)}\right) = \left(Enc_{pk}\left(m^{(1)}\right), \dots, Enc_{pk}\left(m^{p(n)}\right)\right)$$

Theorem 34. *If* $\Pi = (KeyGen, Enc, Dec)$ *is CPA secure, then for any polynomial* $p\left(n\right)$ *the scheme* $\Pi' = (KeyGen, Enc', Dec')$ *is CPA secure*

Proof Idea. Given an adversary $\mathcal{A}'$ for Π', construct an adversary $\mathcal{A}$ for Π. $\mathcal{A}$ gets one challenge ciphertext, and generates the others on its own.

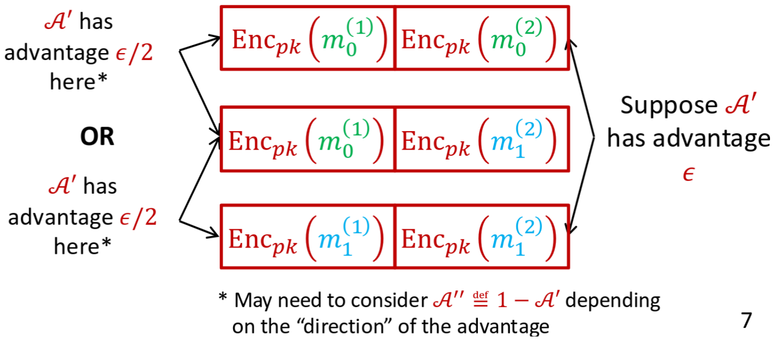

$\mathcal{A}'$ has advantage $\epsilon/2$ here*
OR
$\mathcal{A}'$ has advantage $\epsilon/2$ here*
$\mathrm{Enc}_{pk}\left(m_0^{(1)}\right)$ $\mathrm{Enc}_{pk}\left(m_0^{(2)}\right)$
$\mathrm{Enc}_{pk}\left(m_0^{(1)}\right)$ $\mathrm{Enc}_{pk}\left(m_1^{(2)}\right)$
$\mathrm{Enc}_{pk}\left(m_1^{(1)}\right)$ $\mathrm{Enc}_{pk}\left(m_1^{(2)}\right)$
Suppose $\mathcal{A}'$ has advantage ϵ
* May need to consider $\mathcal{A}'' \overset{\text{def}}{=} 1 - \mathcal{A}'$ depending on the "direction" of the advantage
7

Figure 5: Hybrid argument

So $\mathcal{A}$ can generate the messages $\left(m_0^{(1)}, m_0^{(2)}\right)$, and $\left(m_1^{(1)}, m_1^{(2)}\right)$, and mix the two halves. It can then (for example) provide $\left(m_0^{(1)}, m_0^{(2)}\right)$ and $\left(m_0^{(1)}, m_1^{(2)}\right)$, with an advantage of at least $\frac{\varepsilon}{2}$. Since it can differentiate between the halves, it is then able to differentiate between the messages. □

Now, public key schemes are somewhat inefficient, and slow to compute, but symmetric (private key) schemes are much faster. The solution is called **Hybrid encryption**, where one generates a *session key* k, encrypts it (since it is short) with a public key encryption scheme, and then encrypt m with a private key scheme, using k as the key.

37 Hybrid encryption

Definition 37.1 (Hybrid encryption)**.** *Let* $\Pi = (KeyGen, Enc, Dec)$ *be a public key encryption scheme, and let* (E, D) *be a symmetric key encryption scheme. We can now define a public key encryption scheme* $\Pi' = (Gen', Enc', Dec')$ *, where*

- $Gen' = Gen$
- $Enc'_{pk}(m)$ *samples* $k \leftarrow \{0,1\}^n$*, computes* $c_1 \leftarrow Enc_{pk}(k)$*, and* $c_2 \leftarrow E_k(m)$*. It outputs* $c - (c_1, c_2)$
- $Dec'_{sk}(c_1, c_{"})$*: Compute* $k \leftarrow Dec_{sk}(c_1)$*, and output* $D_k(c_2)$

Theorem 35. *If* Π *is CPA-secure, and* (E, D) *is IND-secure, then* Π' *is CPA secure*

Proof . Let there be $\mathcal{A}'$ for Π'. It receives pk, outputs m_0, m_1, receives $Enc_{pk}(k), E_k(m_b)$, and outputs b'. From this we may construct an adversary $\mathcal{A}$ for Π, or an adversary $\mathcal{B}$ for (E, D). Taking the two messages that $\mathcal{A}'$ outputs, m'_0, m'_1, then we may consider the following:

1. $(Enc_{pk}(k), E_k(m'_0))$
2. $(Enc_{pk}(0^n), E_k(m'_0))$
3. $(Enc_{pk}(0^n), E_k(m'_1))$
4. $(Enc_{pk}(k), E_k(m'_1))$

From 1 - 4 $\mathcal{A}'$ let us say that $\mathcal{A}'$ has the advantage ε. As a result, there are 2 stages where it has an advantage of at least $\frac{\varepsilon}{3}$. We can now build $\mathcal{A}$, that simulates $\mathcal{A}'$. Upon receiving m'_0, m'_1, we will output

$$m_0 = k$$
$$m_1 = 0^n$$

and receive in return the public key encryption of one of them $c^* = Enc_{pk}(m_b)$. We can then then return $(c^*, E_k(m'_0))$ to $\mathcal{A}'$, which will output the correct b' with its advantage, since it can differentiate between $Enc_{pk}(k)$ and $Enc_{pk}(0^n)$. Conversely, if we set

$$m_0 = m'_0 m_1 = m'_1 \tag{46}$$

Then we can give $\mathcal{A}'$ the input $(Enc_{pk}(0^n), c^*)$, where $c^* = E_k(m_b)$, and since it has the advantage for $E_k(m'_0)$ and $E_k(m'_1)$, we will once again get back the correct b'.
Finally, we can also have $\mathcal{A}$ output

$$m_0 = 0^n$$
$$m_1 = k$$

Receiving back $c^* = Enc_{pk}(m_b)$, give $(c^*, E_k(m'_1))$ to $\mathcal{A}'$, and thanks to the advantage that $\mathcal{A}'s$ has between $Enc_{pk}(0^n)$ and $Enc_{pk}(k)$, it will return the correct b'. □

37.1 El-Gamal Encryption

Behold a real public key encryption scheme. It is based on Diffie-Hellman key agreement, and relies on the DDH assumption. Recall the DDH assumption: Let $\mathcal{G}$ be a PPT algorithm that on input 1^n, outputs $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q, that is generated by g, and q is an $n$bit prime.

Definition 37.2 (The Decisional Diffie Hellman (DDH) Assumption)**.** *For every PPT algoirithm* $\mathcal{A}$ *there exists a negligible function* $v(\cdot)$ *such that*

$$\left|\Pr\left[\mathcal{A}\left(\mathbb{G}, q, g, g^x, g^y, g^{xz}\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathbb{G}, q, g, g^x, g^y, g^z\right) = 1\right]\right| \leq v(n)$$

Where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$*, and* $x, y, z \leftarrow \mathbb{Z}_q$

So: Let $\mathcal{G}$ be a PPT algorithm that on input 1^n outputs $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of order q, that is generated by g. We will define a public key encryption scheme $\Pi = (KeyGen, Enc, Dec)$ as

- $Gen'(1^n)$: Sample $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$, and $x \leftarrow \mathbb{Z}_q$. Let $h = g^x$, and output $pk = (\mathbb{G}, q, g, h)$, and $sk = x$
- $Enc_{pk}(m)$: Sample $y \leftarrow \mathbb{Z}_q$, and output $c = (g^y, h^y \cdot m)$
- $Dec_{sk}(c_1, c_2)$: Output $m = \dfrac{c_2}{c_1^x}$

Note that there are methods for encoding binary strings as group elements, and that for simplicity, we assume that the plaintext set is $\mathbb{G}$.

$$Dec_{sk}\left(Enc_{pk}\left(m\right)\right) = Dec_{sk}\left(g^y, h^y \cdot m\right) = \frac{h^y \cdot m}{\left(g^y\right) tox} = \frac{\left(g^x\right)^y \cdot m}{\left(g^y\right)^x} = m$$

Theorem 36 (Security)**.** *Under the DDH assumption, the scheme* Π *is secure*

Proof . Hey look! Another reduction. We will assume that there exists $\mathcal{A}$ that breaks Π, and so we can build $\mathcal{D}$ that breaks DDH. So, $\mathcal{D}$ receives (g^x, g^y, g^z), and needs to return if $z = xy$, or if z is random.

Let us construct $\mathcal{D}$, that receives $(\mathbb{G}, q, g, g_1, g_2, g_3)$. It will give $\mathcal{A}$ $pk = (\mathbb{G}, q, g, g_1)$. $\mathcal{A}$ will return m_0, m_1, and then $\mathcal{D}$ will return to $\mathcal{A}$ the ciphertext $c^* = (g_2, g_3 \cdot m_b)$.

Case I:

$$(g, g_1, g_2, g_3) = (g, g^x, g^y, g^{xy})$$

Here, $\mathcal{A}$'s view is identical to the CPA experiment, and therefore

$$\Pr\left[\mathcal{D}\left(\mathbb{G}, q, g, g^x, g^y, g^{xy}\right) = 1\right] = \Pr\left[IND^{CPA}_{\Pi,\mathcal{A}}\left(n\right) = 1\right]$$

Case II: Let z be a random number, of appropriate size:

$$(g, g_1, g_2, g_3) = (g, g^x, g^y, g^z)$$

The view of $\mathcal{A}$ is independent of the bit b, and so

$$\Pr\left[\mathcal{D}\left(\mathbb{G}, q, g, g^x, g^y, g^z\right) = 1\right] = \frac{1}{2}$$

So, given $(\mathbb{G}, q, g, g_1, g_2, g_3)$, our algorithm $\mathcal{D}$ will generate $pk = (\mathbb{G}, q, g, g_1)$, and upon $\mathcal{A}$ response of m_0, m_1, it will return $c^* = (g_2, g_3 \cdot m_b)$. It will take $\mathcal{A}$'s output b', and output 1 if $b' = b$, and 0 otherwise. By the DDH assumption:

$$\begin{aligned} v\left(n\right) &\geq \left|\Pr\left[\mathcal{D}\left(\mathbb{G}, q, g, g^x, g^y, g^{xy}\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathbb{G}, q, g, g^x, g^y, g^z\right) = 1\right]\right| \\ &= \left|\Pr\left[IND^{CPA}_{\Pi,\mathcal{A}}\left(n\right) = 1\right] - \frac{1}{2}\right| \end{aligned}$$

□

37.2 RSA encryption

37.2.1 The RSA assumption

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d), where p, q are n bit primes, $N = pq$, $gcd\left(e, \phi\left(N\right)\right) = 1$, and $d = e^{-1} \mod \phi\left(N\right)$. Here $\phi\left(N\right)$ is the order of our set $\mathbb{Z}^*_{N=pq}$, such that $\phi\left(N\right) = \left(p-1\right)\left(q-1\right)$.

Definition 37.3 (RSA Assumption)**.** *For every PPT* $\mathcal{A}$ *there exists a negligible function* $v\left(\cdot\right)$ *such that*

$$\Pr\left[\mathcal{A}\left(N, e, x^e \mod N\right) = x\right] \leq v\left(n\right)$$

Where $(N, e, d) \leftarrow GenRSA\left(1^n\right)$ *and* $x \leftarrow \mathbb{Z}^*_N$.

In short, given $x^e \mod N$, it is very hard to compute x. In other words, $f_e\left(x\right) = x^e \mod N$ is a one way permutation family. f_d is the inverse of f_e, since $ed = 1 \mod \phi\left(N\right)$

37.2.2 Textbook RSA encryption

Let GenRSA be a PPT algorithm that on input 1^n outputs (N, e, d), where p, q are n bit primes, $N = pq$, $gcd\left(e, \phi\left(N\right)\right) = 1$, and $d = e^{-1} \mod \phi\left(N\right)$.

From here we create the public key $pk = (N, e)$, and the private key $sk = d$. Thus:

$$Enc_{pk}\left(m\right) = m^e \mod N \quad (47)$$

$$Dec_{sk}\left(c\right) = c^d \mod N \quad (48)$$

This is... Not a great system to be honest. It was first suggested in 1977, but the security definitions, like CPA-security were created in 1982. We may firstly note that Enc is deterministic, which is immediately **bad**. We will note that it is also not CPA-secure, many attacks are known. For example, if $m^e < N$, then $c = [m^e \mod N] = m^e$, and so $c^{\frac{1}{e}} = m$.

We may take this moment to state **NEVER USE TEXTBOOK RSA!!!**
To emphasise this point I have used atrocious grammar. I hope this helps you remember this.
(There is in fact an argument, which I will link `here`, that RSA should never be used. It is not a part of the course, and I have added it purely for your own interest.)

37.2.3 PKCS

Version 1.5 was standard issued by RSA labs in 1993. The idea is random padding:

$$pk = (N, e) \tag{49}$$
$$sk = d \tag{50}$$

So, $Enc_{pk}(m) = (r \| m)^e \mod N$, for a freshly chosen random r. This has the drawbacks that no proof of CPA security exists (aside from if m is very short). Chosen plaintext attacks are known if r is too short, and chosen ciphertext attacks are also known. In short, we do not know if it is secure, nor do we even have a neat little assumption (like DDH) that *if* it holds, we know it to be secure.

Next is version 2.0, which uses a more structured padding: Optimal Asymmetric Encryption Padding (OAEP). OAEP introduces redundancy, so that not every $c \in \mathbb{Z}_n^*$ is a valid ciphertext. This means that $Dec_{sk}(\cdot)$ must check for proper formatting upon decryption, and reject if it does not exist. This can be proved to be CCA-secure under the RSA assumption, if G and H are modelled as "random" hash functions. It is widely used in practice.

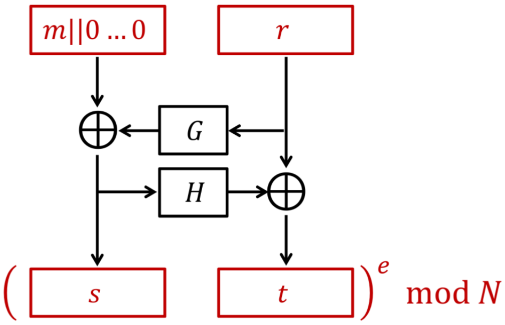


Figure 6:

Note that the RSA permutation family is **not** a CPA-secure PKE scheme. It is however a family of *trapdoor permutations*, which are one way permutations that may be efficiently inverted using a trapdoor. We will show a generic construction of a CPA secure scheme from any TDP (trapdoor permutation) family:

Definition 37.4 (Trapdoor permutation family). *A tuple $(Gen, Samp, f, f^{-1})$ of PPT algorithms is a **trapdoor permutation family** if:*

- *$Gen(1^n)$ outputs pairs (I, td) defining a domain $\mathcal{D}_I$*
- *$(Gen_1, Samp, f)$ is a one way permutation family, where Gen_1 is obtained from Gen by outputting only I*
- *f^{-1} is deterministic, and for all (I, td) and $x \in \mathcal{D}_I$ it holds that $f_{td}^{-1}(f_I(x)) = x$*

For simplicity, we will typically write $x \leftarrow \mathcal{D}_I$ instead of $x \leftarrow \text{Samp}_I$, and (Gen, f, f^{-1}) instead of $(\text{Gen}, \text{Samp}, f, f^{-1})$. To summarise our cryptographic primitives once more:

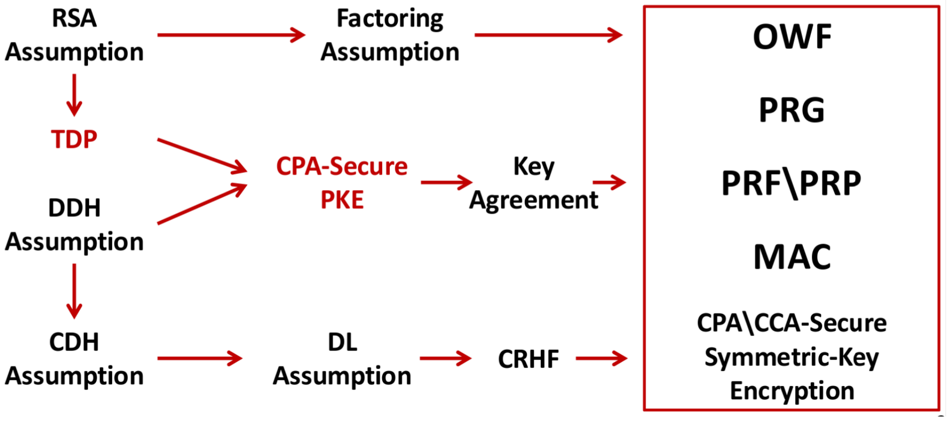


Figure 7: Cryptographic primitives

38 Constructions

Part XI

Lecture 8 — 2025-12-31

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

39 Digital signatures

Alice and Bob wish to communicate, but Eve completely controls the channel. We would like to assure the receiver of a message that it has not been modified. We will discuss the public key counterpart of message authentication codes. The signer holds a *secret* signing key, and the verifier knows the corresponding public *verification* key. This means that anyone can verify the signature, but only one person can create it. This is the inverse of encryption, where everyone knows the public encryption key, but only 1 person knows the private decryption key.

This has the syntax $\Pi = (Gen, Sign, \text{Vrfy})$:

- The key generation algorithm Gen on input 1^n outputs a signing key sk, and a verification key vk
- Sign takes a signing key sk, and a message m, and outputs a signature σ
- Vrfy takes a verification key vk, a message m, and a signature σ, and outputs a bit b

Correctness: For every message m

$$\Pr\left[\text{Vrfy}_{vk}\left(m, \text{Sign}_{sk}\left(m\right)\right) = 1\right] = 1$$

To compare against MACs:

Signatures	MACs
n users require only n secret keys	n users require n^2 secret
Same signature can be verified by all users	
Publicly verifiable and transferable	Privately verifiable and non transferable
Provides non repudiation	More efficient (2 - 3 orders of magnitude faster)

Table 1:

39.1 Security of Signatures

$\mathcal{A}$ knows vk, and can adaptively ask for signatures of messages of its choice. It then tries to forge a signature on a new message.

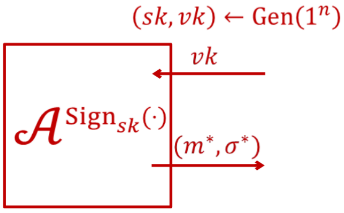

$(sk, vk) \leftarrow \text{Gen}(1^n)$
vk
$\mathcal{A}^{\text{Sign}_{sk}(\cdot)}$
(m^*, σ^*)

Figure 8: Signature game

We finish with Q, the set of all queries asked by $\mathcal{A}$, and

$$\text{SigForge}_{\Pi,\mathcal{A}}\left(n\right) = \begin{cases} 1, & \text{if } \text{Vrfy}_{vk}\left(m^*, \sigma^*\right) = 1 \wedge m^* \notin Q \\ 0, & \text{else} \end{cases}$$

Definition 39.1. Π *is* ***existentially unforgeable against an adaptive chosen message attack*** *if for every PPT adversary $\mathcal{A}$, there exists a negligible function $v\left(\cdot\right)$ such that*

$$\Pr\left[SigForge_{\Pi,\mathcal{A}}\left(n\right) = 1\right] \leq v\left(n\right)$$

40 Constructions

One time signatures are used to construct stateful signatures, which may then be used to construct stateless signatures.

40.1 One time signatures

We will demonstrate **Lamport's One time scheme**:

- $sk = (x_0, x_1)$
- $vk = (f(x_0), f(x_1))$, where f is a one way function
- $\text{Sign}_{sk}(b) = x_b$
- Vrfy receives the message, and the signature, and checks it against the relevant side of the verification key

This way $\mathcal{A}$ needs to compute x_{1-b}, which is equivalent to computing the inverse of f.

More formally: Let f be an OWF. we will define a signature scheme $\Pi = (Gen, \text{Sign}, \text{Vrfy})$ for 1 bit messages as follows:

- Gen (1^n): Sample $x_0, x_1 \leftarrow \{0,1\}^n$, and compute $y_0 = f(x_0)$, and $y_1 = f(x_1)$, output $sk = (x_0, x_1)$ and $vk = (y_0, y_1)$
- $\text{Sign}_{sk}(b)$: Output $\sigma = x_b$
- $\text{Vrfy}_{vk}(b, \sigma)$: If $f(\sigma) = y_b$ output 1, otherwise output 0

Theorem 37. *If f is an OWF, then Π is a secure one time signature scheme for 1 bit messages*

Proof . The concept is that $\mathcal{A}$ forges a signature on $b^* \implies \mathcal{A}$, inverts $y_{b^*} = f(x_{b^*})$. Inverting $f(x_{b^*})$ is clearly hard, even when given x_{1-b^*} and $f(x_{1-b^*})$. An inverter can guess the forged bit b^* ahead of time with probability $\frac{1}{2}$.

We can construct an inverter $\mathcal{B}$ as follows, which takes as input $y = f(x)$ for some $x \leftarrow \{0,1\}^n$.

1. Choose $b^* \leftarrow \{0,1\}$ and set $y_{b^*} = y$
2. Sample $x_{1-b^*} \leftarrow \{0,1\}^n$, and set $y_{1-b^*} = f(x_{1-b^*})$
3. Run $\mathcal{A}$ on input $vk = (y_0, y_1)$
4. When $\mathcal{A}$ requests a signature on b:
   - If $b = b^*$ abort
   - If $b = 1 - b^*$ output x_{1-b^*}
5. If $\mathcal{A}$ output a forgery σ^* on b^*, output σ^*

□

So

$$\begin{aligned}\Pr\left[\mathcal{B}(f(x)) \in f^{-1}(f(x))\right] &\geq \Pr\left[\text{SigForge}_{\Pi,\mathcal{A}}(n) = 1 \wedge \mathcal{B} \text{ does not abort}\right] \\ &= \Pr\left[\text{SigForge}_{\Pi,\mathcal{A}}(n) = 1\right] \cdot \Pr[\mathcal{B} \text{ does not abort}] \\ &\Pr\left[\text{SigForge}_{\Pi,\mathcal{A}}(n) = 1\right] \cdot \frac{1}{2}\end{aligned}$$

We may note that this scheme only works one time, for a single bit. We may extend this to l bit messages by creating

$$sk = \begin{bmatrix} x_0^1 & x_0^2 & \dots & x_0^l \\ x_1^1 & x_1^2 & \dots & x_1^l \end{bmatrix} \tag{51}$$

$$vf = \begin{bmatrix} f\left(x_0^0\right) & \dots & f\left(x_0^l\right) \\ f\left(x_1^0\right) & \dots & f\left(x_1^l\right) \end{bmatrix} \tag{52}$$

Or formally: Let f be an OWF. We define the signature scheme $\Pi = (Gen, \text{Sign}, \text{Vrfy})$ for l bit messages as follows:

- Gen (1^n): For each $i \in [l]$, and $b \in \{0,1\}$, sample $x_{i,b} \leftarrow \{0,1\}^n$ and compute $y_{i,b} = f(x_{i,b})$. Output $sk = \{(x_{i,0}, x_{i,1})\}_{i \in [l]}$ and $vk = \{(y_{i,0}, y_{i,1})\}_{i \in [l]}$
- $\text{Sign}_{sk}(m = m_1 \dots m_l)$: Outputs $\sigma = (x_{1,m_1}, \dots, x_{l,m_l})$
- $\text{Vrfy}_{vk}(m = m_1 \dots m_l, \sigma = (x_1, \dots, x_l))$: If $\forall i \in [l]\ \ f(x_i) = y_{i,m_i}$ output 1, else 0

Theorem 38. *If f is an OWF, then Π is a secure, one time signature scheme for l bit messages*

Proof Idea. Suppose that $\mathcal{A}$ asks for a signature on m, and then forges on $m^* \neq m$. The inverter $\mathcal{B}$ needs to guess $i \in [l]$ such that $m_i^* \neq m_i$ as well as guess the bit m_i^*. □

40.1.1 Summary

Lamport theorised in 1979 that if OWFs exist, then for any polynomial $l = l(n)$ there is a one time signature scheme for signing l bit messages. The following theorem is known as the *Hash and Sign* paradigm:

Theorem 39. *If CRHFs exist, then there is a one time signature scheme that can sign messages of arbitrary polynomial length.*

40.2 Stateful signatures

We are now extending the game, such that $\mathcal{A}$ may request the signature of many different messages.
The signer updates the signing key after each signature.

- The initial state sk_1 is produced by Gen : $(vk, sk_1) \leftarrow Gen(1^n)$
- Signing the ith message updates sk_i to $sk_{i+1} : (\sigma, sk_{i+1}) \leftarrow \text{Sign}_{sk_i}(m_i)$
- Verification requires only vk

For existential unforgeability against an adaptive chosen message attacks

- $\mathcal{A}$ knows vk, and can adaptively ask for signatures of its choice
- The signing oracle maintains the internal state sk_i
- $\mathcal{A}$ tries to forge a signature on a new message

Let us create a stateful scheme. Let $\Pi = (Gen, \text{Sign}, \text{Vrfy})$ be a one time signature scheme for signing "sufficiently long" messages. for $m = m_1 \ldots m_n \in \{0,1\}^n$, we let $m\mid_i \overset{def}{=} m_1 \ldots m_i$ (and $m\mid_0 \overset{def}{=} \varepsilon$).
We will define $\Pi' = (Gen', \text{Sign}', \text{Vrfy}')$ for signing n bit messages as follows:

- The signer's state is a binary tree with 2^n leaves
- Each node $w \in \{0,q\}^{<n}$ has a left child $w0$, and a right child $w1$
- The tree is of exponential size, but is never fully constructed

Key generation: Each node $w \in \{0,1\}^{\leq n}$ is associated with $(vk_w, sk_w) \leftarrow Gen(1^n)$. Keys are generated, and stored only when needed. The state sk'_i consists of all keys and signatures that were generated so far. $vk' = vk_\varepsilon$ and $sk'_1 = sk_\varepsilon$. Note that vk_ε is the root node, with children vk_0, vk_1.

To sign a message $m \in \{0,1\}^n$:

1. Generate a path from the root, to the leaf labelled m: For each proper prefix w of m sample $(vk_{w0}, sk_{w0}), (vk_{w1}, sk_{w1}) \leftarrow Gen(1^n)$
2. Certify the path: For each proper prefix w of m, compute $\sigma_w = \text{Sign}_{sk_w}(vk_{w0}, vk_{w1})$
3. Compute $\sigma_m = \text{Sign}_{sk_m}(m)$
4. Store all generated keys as part of the updated state
5. Output the signature $\left(\left\{\sigma_{m|_i}, vk_{m|_i0}\right\}_{i=1}^{n-1}, \sigma_m\right)$

Simple example: The message $m = 111$ receives the signature $\text{Sign}_{sk111}(111), \text{Sign}_{sk11}(vk_{111}), \text{Sign}_{sk1}(vk_{11}), \text{Sign}_{sk\varepsilon}(vk_1)$. This simple example is missing the fact that if we now want to sign 110, we need to resign 11, which is a problem from the attack scheme. To fix this, each parent provides the signature for *both* its children at once, and we thus avoid this issue.

Theorem 40. *If* Π *is a one time signature scheme, then* Π' *is existentially unforgeable against chosen message attacks*

Proof Idea. Each sk_w is used to sign exactly one "message". If w is an internal node, then sk_w is used to sign (vk_{w0}, vk_{w1}), and if w is a leaf then sk_w is used to sign w. □

Proof Idea #2. Suppose that $\mathcal{A}$ asks to forge a signature $\left(\left\{\sigma^*_{m^*|_i}, vk^*_{m^*|_i0}, vk^*_{m^*|_i1}\right\}_{i=0}^{n-1}, \sigma^*_{m^*}\right)$ on m^*. There are two possible cases:

1. The full path to the leaf m^* already existed, and $\mathcal{A}$ used the same path. This implies that $\mathcal{A}$ must have forged a signature that is a relative of vk_{m^*}, and did not receive any signature that is a relative of vk_{m^*}
2. The full path to leaf m^* did not exist, or $\mathcal{A}$ used a different path. This implies that $\mathcal{A}$ must have forged a signature that is a relative of $vk_{m^*|_i}$ for $i \in \{0, \ldots, n-1\}$, and received exactly one signature that is a relative of $vk_{m^*|_i}$

□

This has the problem of needing to remember **all** the sks, since once we have signed a message, we cannot use sk_ε any more, which is necessary to sign another message. We can now move on to stateless signatures, and thus remove this need for state.

40.3 Stateless signatures

Instead of remembering sk_i at every stage, we use PRFs to create them on the fly. The signer's secret key sk is a seed for a PRF $F_{sk}(\cdot)$. $(r_w, r'_w) \overset{def}{=} F_{sk}(w)$ is used as the randomness needed for each node $w \in \{0,1\}^{\leq n}$:

- If $w \in \{0,1\}^{<n}$ then r_w is used for sampling (vk_w, sk_w) and r'_w is used for signing (vk_{w0}, vk_{w1})
- If $w \in \{0,1\}^n$, then r_w is used for sampling (vk_w, sk_w), and r'_w is used for signing w

Theorem 41. *If Π is a one time signature scheme, and F is a PRF, then Π'' is existentially unforgeable against chosen message attacks*

Proof Idea. Any adversary $\mathcal{A}$ against Π'' can be used either as an adversary against the stateful scheme Π', or as a distinguisher against the PRF F

$$\begin{aligned}\Pr\left[\text{SigForge}_{\Pi'',\mathcal{A}}(n)=1\right] &\leq \left|\Pr\left[\text{SigForge}_{\Pi'',\mathcal{A}}(n)=1\right]-\Pr\left[\text{SigForge}_{\Pi',\mathcal{A}}(n)=1\right]\right|+\Pr\left[\text{SigForge}_{\Pi',\mathcal{A}}(n)=1\right]\\ &=\left|\Pr\left[\mathcal{D}^{F_{sk}(\cdot)}(1^n)=1\right]-\Pr\left[\mathcal{D}^{f(\cdot)}(1^n)=1\right]\right|+\Pr\left[\text{SigForge}_{\Pi',\mathcal{A}}(n)=1\right]\end{aligned}$$

□

41 Certificates and public key infrastructure

Public key cryptography is great, but we need to distribute the public keys *somehow.* Keys must be authenticated in order to avoid man in the middle attacks. This is done by making use of **Certificate Authorities**:

- A **certificate** is a signature binding an identity to a public key
- We assume that we already trust the CA's verification key vk_{CA} (by hard wiring it into the browser source code or some such)
- The CA provides Alice with $cert_{CA\to A} \overset{def}{=} \text{Sign}_{sk_{CA}}$ (Alice's key is pk_A)
- Alice then sends to Bob both pk_A, and $cert_{CA\to A}$

So for example, we can have a root, that has signed all of HUJI, www.gov.il, and CNN. HUJI then signs on CS, and Chem, and CS can sign on Alice, and Bob. This way, everyone only signs a small number of relevant keys, and I can use this chain of trust to trust someone else's key, because I trust the root node.
Certificates should not be valid indefinitely, since an employee may leave or get fired, and secret keys can get stolen. One solution is to add an expiration date, such that the signature is not valid after that date, another approach is to add a revocation list that the authority publishes, and when I received a signed key, I check it against the CA's revoked list.

42 User-server identification

We need a way to identify users to websites, like when you log in to moodle. A trivial method would be for the user to hold a password p, the server to know $y = f(p)$ for some function f, and the user identifies themselves by sending p. This is obviously terrible. It can however be *slightly* improved by using a signature scheme. The user has the signing key sk, and the server knows the verification key vk. The user identifies themselves by signing a message that the server has randomly generated for them.

Part XII

Exam 2025A — 2026-01-07

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

43 Question 1

Let there be a family of collision resistant functions $H_s \{0,1\}^{2n} \to \{0,1\}^n$. Let there also be a PRG $G : \{0,1\}^{n+1} \to \{0,1\}^{2n}$.

43.1 Part A

Consider

$$F_s(x_1 \| x_2) = H_s(H_s(x_1) \| H_s(x_2))$$

Where $|x_1|, |x_2| = 2n$, and $F_s : \{0,1\}^{4n} \to \{0,1\}^n$. Is the family F_s collision resistant?

Sol. Yes. Let us assume towards contradiction that F_s is not collision resistant, so there exists an adversary $\mathcal{A}$ that finds collisions in F_s. Let us create $\mathcal{B}(s)$, which runs $\mathcal{A}(s)$, which returns $(x_1, x_2), (x_1', x_2')$. Since $(x_1, x_2) \neq (x_1', x_2')$ then at least one of the pair of variables $x_i, x_i' : i \in \{1,2\}$ are different, so let us assume wlog that $x_1 \neq x_1'$. If so, then there are 2 cases:

1. $H_s(x_1) = H_s(x_1')$ in which case we are done, and return (x_1, x_1')
2. $H_s(x_1) \neq H_s(x_1')$ in which case we return $(H_s(x_1) \| H_s(x_2) \| H_s(x_1') \| H_s(x_2'))$

43.2 Part B

Consider

$$L_s(x) = H_s(G(x))$$

Where

$$L_s : \{0,1\}^{n+1} \to \{0,1\}^n$$

Is the family L_s collision resistant?

Sol. No. We will bring a counterexample of (H_s', G') such that L_s is not collision resistant.

$$H_s' = H_s \tag{53}$$

$$G'(x) = \begin{cases} G(x), & \text{if } x \notin \{0^n, 1^n\} \\ 0, & \text{if } x \in \{0^n, 1^n\} \end{cases} \tag{54}$$

Theorem 42 (Claim 1)**.** *G' is a PRG*

Proof . We will assume towards contradiction that there exists an adversary $\mathcal{A}$ that can differentiate between the output of G' and random, and from that build the adversary $\mathcal{B}$ that can differentiate between the output of G and random. It will be exactly the same adversary, and will have the same probability as G for differentiating between G and random, with the addition of $\frac{2}{2^n}$. Since finding this collision in G is in fact negligible, and the addition of $\frac{2}{2^n}$ is also negligible, then the finding of this collision is also in fact negligible. □

Theorem 43 (Claim 2)**.** *$L_s(x) = H_s(G'(x))$ is not a CRH*

Proof . Pretty trivial, since we know the definition of G', and may simply give L_s the two inputs such that G returns the same output, and we have found a non trivial collision in L_s □

44 Question 2

Let $f : \{0,1\}^n \to \{90,1\}^n$ be a one way function. We will use this to create a new signature scheme $\Pi = (Gen, \text{Sign}, \text{Vrfy})$:

- Gen $(1^n) = x_1, \ldots, x_n \leftarrow \{0,1\}^n$, $sk = (x_1, \ldots, x_n)$, $vk = (y_1, \ldots, y_n) = (f(x_1), \ldots, f(x_n))$
- Sign (sk, m): $\sigma = \bot$ (as in, empty string). For $1 \leq i \leq n$, if $m[j] = 1$, then $\sigma \| x_i$, then return σ
- Vrfy (vk, m): Passes over every bit in the message, and knows that the corresponding part of the message must be the preimage of a part of the vk, so it computes the function of it, and checks if it appears in the verification key

44.1 Part A

Show that the system is not secure as a one time signature.

Sol. It's trivial. The empty message 0^n will have the signature $\perp$, without even calling the oracle.

44.2 Part B

Correct the signature scheme such that it is now secure, and that the size of vk is $n^2 + n(\log(n) + 1)$ bits.

Sol. We will note that in the unaltered scheme, an adversary can change an arbitrary 1 in the message to a 0, by simply removing the relevant part of the signature. We can resolve this by signing the number of 0s in the message, which requires $\log(n) + 1$ bits, and thus the adversary cannot change the numbers of 0s, since he would also have to change the signature of the number of 0s:

- KeyGen (1^n): $sk = \left(x_1, \ldots, x_n, x^0_{n+1}, \ldots, x^0_{n+\log n+1}, x^1_{n+1}, \ldots, x^1_{n+\log n+1}\right)$
  $vk = \left(f(x_1), \ldots, f(x_n), f\left(x^0_{n+1}\right), \ldots, f\left(x^0_{n+\log n+1}\right), f\left(x^1_{n+1}\right), \ldots, f\left(x^1_{n+\log n+1}\right)\right)$
- Sign (sk, m): Sign $(m) \|$Lamport (zeroes (m))

This solves it in $n^2 + 2n(\log(n) + 1)$.
To prove it, let us assume towards contradiction that there exists adversary $\mathcal{A}$ that can win the game against this scheme. So, $\mathcal{A}$ outputs m, and receives in return from the oracle $\text{Sign}(m), \text{Lamport}(\text{zeroes}(m))$, and then at the end outputs $m^*, \text{sign}(m^*), \text{Lamport}(\text{zeroes}(m^*))$. There are now two cases:

1. Zeroes (m^*) = Zeroes (m): Then this message must be a permutation of another, since there are the same number of 0s. In this case, then we may break it similarly to how we did Lamport.
2. Zeroes $(m^*) \neq$ Zeroes (m): In this case, then we have succeeded, since we have created a new message with the same signature.

In order to remove the 2, then we may simply change KeyGen to remove the doubling of the bits from $x_{n+1}, \ldots, x_{n+\log n+1}$, and Sign to be $\text{Sign}(m\|\text{zeroes}(m))$. This may be proven with the exact same proof.

45 Question 3

45.1 Part A

Given a cyclic group (G, g, q) such that DDH holds $((g^x, g^y, g^{xy}) \approx (g^x, g^y, g^z))$, let there be two distributions:

$$\left(g^{a_1}, g^{a_2}, g^{a_1 b}, g^{a_2 b}\right) \tag{55}$$

$$\left(g^{a_1}, g^{a_2}, g^{r_1}, g^{r_2}\right) \tag{56}$$

Such that $a_1, a_2, r_1, r_2, b \leftarrow \mathbb{Z}_q$. Show that these distributions are indistinguishable.

Sol. Let us assume towards contradiction that they are distinguishable. So, we are building $\mathcal{A}(g^x, g^y, T)$ where $g \leftarrow \{g^{xy}, g^z\}$ that succeeds against DDH. We will do this by building $\mathcal{B}(g^x, g^{a_2}, T, (g^y)^{a_2})$. When T is random, then $\mathcal{B}$ has received lower option, and when T is g^{xy}, then $\mathcal{B}$ has received the top option. We have thus built an adversary that may win DDH.

45.2 Part B

We will define a key exchange protocol. In order for Alice and Bob to swap keys, Alice chooses $k, r \leftarrow \{0,1\}^n$, and sends Bob $s = k \oplus r$. Bob chooses $t \leftarrow \{0,1\}^n$, and sends $u = s \oplus t$. Alice sends Bob $w = u \oplus r$. Alice outputs k, and Bob outputs $w \oplus t$. Show that the protocol is correct, and whether or not it is secure.

Sol. Correctness:

$$\begin{aligned} w \oplus t &= u \oplus r \oplus t \\ &= s \oplus t \oplus r \oplus t \\ &= k \oplus r \oplus t \oplus r \oplus t \\ &= k \end{aligned}$$

Security: Not secure, in the slightest. The adversary observes $s = k \oplus r$, and $u = s \oplus t$. From this, they may compute $s \oplus u = k \oplus r \oplus k \oplus r \oplus t = t$. From there, like B, they have t, and when w is transmitted, they may compute $w \oplus t = k$, and find the secret key.

Part XIII

Lecture 9 — 2026-01-14

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

46 Introduction

Zero knowledge proofs are proofs that reveal nothing beyond the validity of the assertion being proved. This seems to be a contradicting definition, but we will see that it is not. They were introduced by Goldwasser, Micali, and Rackoff in 1985, and are a central tool in cryptographic protocols. We are not going to pass over the entire powerpoint presentation, this is simply going to be an introductory lecture.

Consider for example proving that I have 1000NIS in my pocket. I could prove by taking it out and counting it, but I could potentially want to prove it without showing you what types of notes I have.

A less contrived example would be finding a mathematical proof, which also has monetary value. I do not want to publish this proof, since then I no longer have the monetary value of the secret (suppose it is an algorithm, which you do not want everyone else to be able to implement). So, zero knowledge proofs can be helpful here, to show that I have proven the theorem, but not distribute the solution to everybody.

This is often thought of as an interactive process (it does not have to be, but it can be easier to think of it as such, and so we will begin by doing so).

A classic example is proving that you have solved a sudoku, without demonstrating the result. This is classic, and there are papers that demonstrate it, that are supposedly very readable, and easy to understand.

Sudoku is a slightly contrived example, since it is very clearly in P, consider instead the 3 colour problem. Can Alice prove that the graph is 3 colourable without revealing the 3 colouring? Well, hopefully we will discuss that later. Furthermore, given $n = pq$, we can show that Alice knows how to factor n, without revealing p or q.

47 Interactive proofs

We will begin by defining classic NP problems. NP is the class of

Definition 47.1 (NP). *All languages L, equipped with an efficiently computable relation R_L such that*

$$x \in L \Leftrightarrow \exists w : (x, w) \in R_L \wedge |w| = \mathit{poly}(|x|)$$

So, given a statement $x \in L$, and proof w, NP proofs are inherently non-ZK, since Bob gains the ability to prove $x \in L$ to others.

Let us extend this with two new ideas:

- Interaction: Replace a static proof with an interactive protocol
- Randomisation: Allow the verifier to toss coins, and err with a small probability

We will create the prover $\mathcal{P}$, which has the random tape $r_{\mathcal{P}}$, and the verifier $\mathcal{V}$, with the random tape $r_{\mathcal{V}}$. We will define the notation:

- $\langle \mathcal{P}, \mathcal{V} \rangle (x)$ is the distribution of the transcript of the protocol
- $\text{out}_{\mathcal{V}} [\langle \mathcal{P}, \mathcal{V} \rangle (x)]$ is the distribution of $\mathcal{V}$'s output

Definition 47.2 (Interactive proof system). *An* ***interactive proof system*** *for a language L is a protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ where $\mathcal{V}$ is computable in probabilistic polynomial time, and the following holds:*

- *Completeness: For every $x \in L$:*
$$\Pr_{r_{\mathcal{P}}, r_{\mathcal{V}}} [\mathit{out}_{\mathcal{V}} [\langle \mathcal{P}, \mathcal{V} \rangle (x)] = \mathit{Accept}] = 1$$
- *Soundness: For every $x \notin L$, and for every computationally unbounded $\mathcal{P}^*$:*
$$\Pr_{r_{\mathcal{P}}, r_{\mathcal{V}}} [\mathit{out}_{\mathcal{V}} [\langle \mathcal{P}^*, \mathcal{V} \rangle (x)] = \mathit{Accept}] \leq \frac{1}{2}$$

We will state that **IP** is the class of all languages with an interactive proof system. IP contains NP, and in fact, $IP = PSPACE$. We can reduce the soundness error from $\frac{1}{2}$ to ε with $\log \left(\frac{1}{\varepsilon}\right)$ independent repetitions.

Definition 47.3 (Isomorphic). *Two graphs $G_0 = (V_0, E_0)$, and $G_1 = (V_1, E_1)$ are* ***isomorphic*** *if there exists a one to one mapping $\pi : V_0 \to V_1$ such that $(u, v) \in E_0 \Leftrightarrow (\pi(u), \pi(v)) \in E_1$ for every $e, v \in V_0$*

That is to say, two graphs are isomorphic if we can rename the nodes in order to transform one into the other. We will note that this problem is in NP, since we can trivially build a verifier, but beyond this, we know nothing. We know not if it is NP-hard, NP-complete, or in P.
We can define the set of isomorphic graphs $GI = \{(G_0, G_1) : G_0 \text{ is isomorphic to } G_1\} \in NP$. Similarly, we can define the other class of graphs that are **not** isomorphic: $GNI = \{(G_0, G_1) : G_0 \textbf{ is not} \text{ isomorphic to } G_1\} \in NP$. This class is not known to be in NP.
We can now ask the question as to how to prove to an efficient verifier that G_0, G_1 are not isomorphic. we will posit that GNI $\in IP$: Given the common input (G_0, G_1), the prover, and the verifier, the verifier will try and find the points where (if) the prover is guessing.
Intuitive solution: The verifier will create two new graphs, which are permutations on the originals, $\pi(G_0), \pi(G_1)$. The verifier will also create $\pi(G_b) : b \leftarrow \{0,1\}$. If the graphs are isomorphic, then all the graphs, including the new ones, are also isomorphic. If they are not, then $\pi(G_0)$ is not isomorphic to $\pi(G_1)$, and $\pi(G_b)$ will be isomorphic to **one** of them.
Formal: The verifier will create $H = \pi(G_b)$ for a random permutation π, and $b \leftarrow \{0,1\}$. H is then sent to the prover. The prover will then find $z \in \{0,1\}$, such that H is isomorphic to G_z, and will then respond with z. The verifier will accept **if and only if** $z = b$.

Theorem 44. *This protocol is an interactive proof for GNI*

Proof . Firstly, we will claim completeness. If $(G_0, G_1) \in$ GNI, then the verifier always accepts, since no graph can be isomorphic to both G_0, and G_1.
Next, soundness, if $(G_0, G_1) \notin$ GNI, then for every $\mathcal{P}^*$ the verifier accepts with probability $\frac{1}{2}$. This is true since $\mathcal{P}^*$'s view is independent of b, so therefore for any $z \in \{0,1\}$ that $\mathcal{P}^*$ will output, we have $\Pr_{b \leftarrow \{0,1\}}[z = b] = \frac{1}{2}$. □

48 Zero knowledge proofs

An interactive proof system is zero-knowledge if whatever can be efficiently computed after interacting with $\mathcal{P}$ on input $x \in L$ can also be computed given only x. This should be true even when $\mathcal{P}$ is interacting with a malicious verifier.

Let us return to the isomorphic classes. Can we prove that G_0 and G_1 are isomorphic without revealing the isomorphism? A solution needs to enable completeness, soundness, and zero knowledge (ZK). Hear me and rejoice, for we are able so to do. Gird thy loins, and behold the following solution:
Intuition: Two graphs are isomorphic if there exists a permutation that transforms from one to the other (π). There therefore also exists π^{-1}, which transforms in the opposite direction. Let us consider another permutation σ, which transforms G_0 to H. Should G_0 and G_1 be isomorphic, then there also exists σ' which transforms from G_1 to H. We can send the transformed graph $H = \sigma(G_0)$ to the prover, and it will respond with the permutation that transforms G_1 to H. This is ZK, since it teaches us nothing on the permutations π, π^{-1}, but demonstrates that the prover can find these permutations.
Formal: The prover will sample a random permutation σ, and send $H = \sigma(G_0)$ to the verifier. The verifier then responds with a request that the prover show that H is isomorphic to G_b, for $b \leftarrow \{0,1\}$. The prover will then respond with

$$\gamma = \begin{cases} \sigma, & \text{if } b = 0 \\ \sigma \circ \pi^{-1}, & \text{if } b = 1 \end{cases}$$

Which provides the required transformation to H for G_b. The verifier then accepts **if and only if** $\gamma(G_b) = H$.
Correctness: If the two graphs are isomorphic, then the verifier will trivially receive what it requested.
ZK: We need to show that the prover did not leak information. Since the only leak that can happen is $\sigma \circ \pi^{-1}$, but since we have further transformed π^{-1} with σ, it acts sort of like a one time pad, and so it does not leak π.
Soundness: The two graphs are not isomorphic, so the prover sends some graph, it can be a transformation of G_0, or G_1, or perhaps some other graph G_5, and in every case, the prover will make a mistake with probability of $\frac{1}{2}$, as required.

49 Zero knowledge proofs for NP

Theorem 45. *Assuming that OWFs exist, then any $L \in NP$ has a zero knowledge interactive proof. Furthermore, the prover's strategy can be implemented in probabilistic polynomial time, provided an NP witness for membership of the ocmmon input.*

Steps. 1. Construct a ZK proof for some NP complete language. Use G3C (graph 3 colouring), and the tool commitments schemes (which are based on OWFs)

2. Given any NP language L, a common input x, and a witness w, we reduce them to G3C, and then use the above ZK proof.

Since we showed reductions in computability, and complexity, we will focus on step one, and commitments schemes □

49.1 Tool: Commitment schemes

These are the basic ingredient in many cryptographic protocols. They are a digital analogue of locked boxes. The sender S has a value v, and a random tape r_S. The sender sends the commit phase to the receiver, and receives a protocol $\langle S, R\rangle$ in response. It sends (v, r_S) in the reveal phase, and the receiver R accepts v **if and only if** (v, r_S) are consistent with the commit phase. Commitment schemes have the following security requirements:

- **Hiding**: At the end of the commit stage, the receiver has no knowledge of v
- **Binding**: The sender cannot find two valid openings (v, r_S) and (v', r'_S) for $v \neq v'$

Definition 49.1 (Hiding)**.** *A commitment scheme $\langle S, R\rangle$ is **computationally hiding** if for every PPT receiver R^* and for every two values $v \neq v'$ it holds that*

$$\mathit{view}_{R^*}\left[\langle S(v), R^*\rangle(1^n)\right] \approx^c \mathit{view}_{R^*}\left[\langle S(v'), R^*\rangle(1^n)\right]$$

*Perfect (statistical) hiding is when $\mathit{view}_{R^*}\left[\langle S(v), R^*\rangle(1^n)\right]$ and $\mathit{view}_{R^*}\left[\langle S(v'), R^*\rangle(1^n)\right]$ are identical (statistically indistinguishable) for any **unbounded** R^**

Definition 49.2 (Binding)**.** *A commitment scheme $\langle S, R\rangle$ is **computationally binding** if for every PPT sender S^* there exists a negligible function $v(n)$ such that*

$$\Pr\left[\left((v, r, v', r'), \mathit{com}\right) \leftarrow \langle S^*, R\rangle(1^n) : v \neq v' \wedge (v, r) \text{ is consistent with com } \wedge (v', r') \text{ is consistent with com}\right]$$

where

$$\mathit{com} \overset{def}{=} \mathit{view}_R\left[\langle S^*, R\rangle(1^n)\right]$$

49.1.1 Some applications of commitments

Consider if you are playing some board game over the phone, which involves throwing dice. Your opponent tells you that they threw 6. Commitments can be used to to ensure that this is the truth. Let us simplify this somewhat into coin flipping, neither party wants to speak first. Alice may send a commitment of what she tossed, Bob then responds with his toss. Alice then sends the reveal for her toss, which Bob can then verify, so this way, Alice could not change her result, and Bob can believe it.

Let us return to ZK proofs for NP.

Definition 49.3. *A graph $G = (V, E)$ is 3 colourable if there exists a mapping $\varphi : V \to \{1, 2, 3\}$ such that $\varphi(u) \neq \varphi(v)$ for every $(u, v) \in E$.*

We want to prove that G is 3 colourable, without revealing a 3 colouring. The high level idea is to break that G is 3-colourable into polynomially many pieces. Each piece does not reveal any information, but combining all the pieces yields a proof that G is 3-colourable. We can implement this using commitments.

Solution: Given the common input $G = (V, E)$, and an auxiliary input to the prover, which is a 3 colouring $\psi : V \to \{1, 2, 3\}$. The protocol is as follows:

- $\mathcal{P}$ uniformly chooses a permutation π over $\{1, 2, 3\}$, and lets $\varphi \overset{def}{=} \pi \circ \psi$
- $\mathcal{P}$ commits to the value $\varphi(w)$ for every $w \in V$ using a statistically binding commitment
- $\mathcal{V}$ uniformly chooses an edge $(u, v) \in E$, and sends it to $\mathcal{P}$
- $\mathcal{P}$ revelas the openings of $\varphi(u)$, and $\varphi(v)$
- $\mathcal{V}$ accepts **if and only if** the openings are valid, i.e. $\varphi(u), \varphi(v) \in \{1, 2, 3\} \wedge \varphi(u) \neq \varphi(v)$

This protocol is repeated $t \cdot |E|$ times for soundness e^{-t}.

Since we have shown ZK proofs for G3C, and there exist reductions for every language in NP, we can thus show a ZK proof for every language in NP.