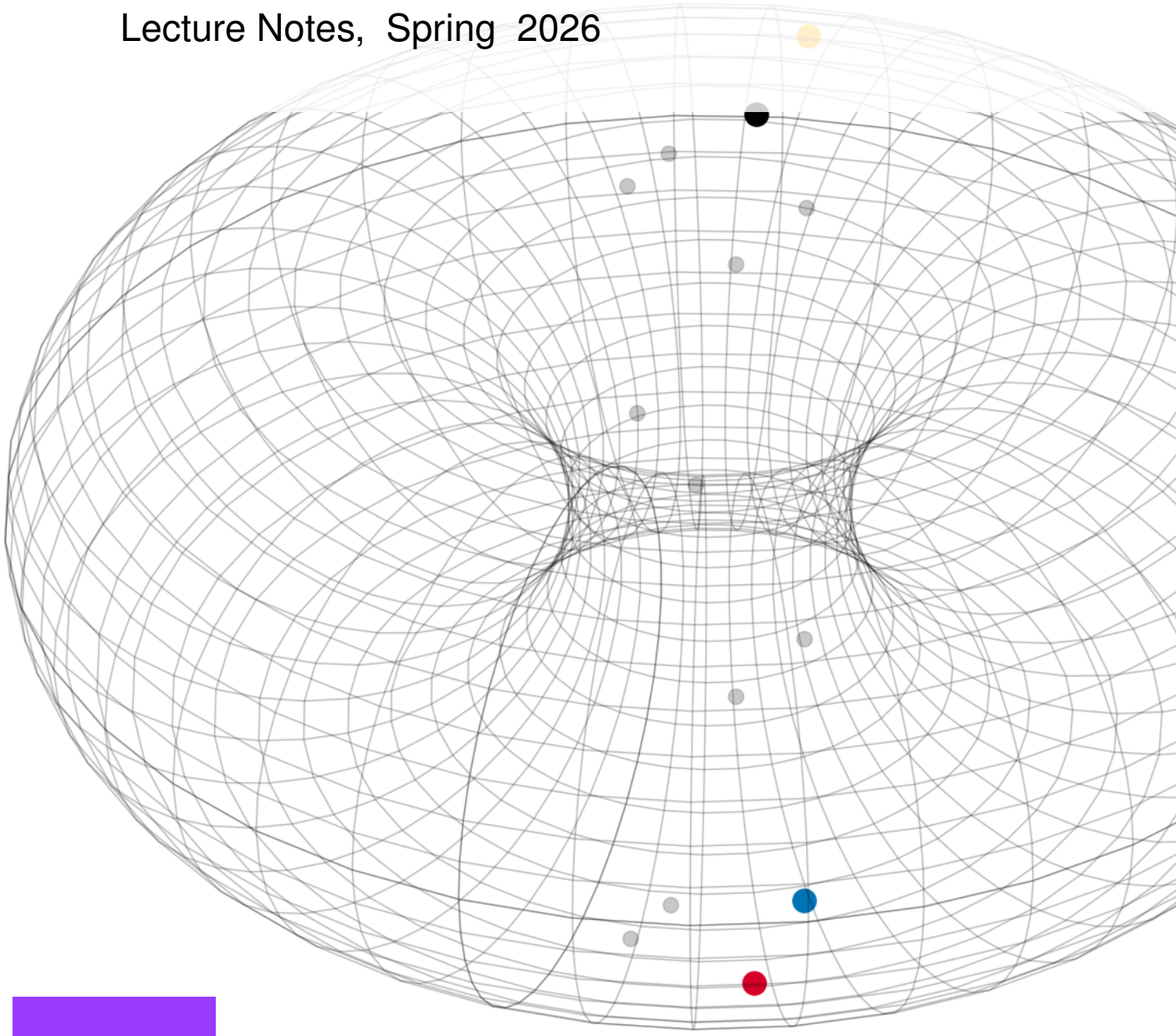
COMP0025: Introduction to Cryptography

Pascal U

Lecture Notes, Spring 2026



Principles of Modern Cryptography

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 6-10 Oct

Overview

What constitutes formally a "break" of a scheme? What powers can an adversary have? When defining security goals and threat models, it is important to provide mathematical proof over some baseline which addresses these questions.

The algorithmic hardness assumptions of certain mathematical problems give the security for cryptographic functions, however this is predicated on a given threat model which fits the use case. Cryptographic proofs must guarantee that no adversary can violate security properties under these stated assumptions. This is a better approach to security than one which is unprincipled, based on heuristics (rules of thumb) or just relying on intuition.

1 Private-Key Encryption Schemes

Definition
A *private-key encryption scheme* Π defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ is a tuple of "efficient" algorithms (Gen, Enc, Dec) with

- Gen : k $\xleftarrow{\$} \mathcal{K}$
- Enc : $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$, $\text{Enc}_k(m) = c$
- Dec : $\mathcal{K} \times \mathcal{C} \to \mathcal{M}$, $\text{Dec}_k(c) = m$

s.t. $\forall m \in \mathcal{M}, \forall k \in \mathcal{K} : \text{Dec}_k(\text{Enc}_k(m)) = m$ (correctness).

2 One-Time Pad

Definition

- $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^\lambda$ for some $\lambda \in \mathbb{N}$
- Gen(1^λ): $k \xleftarrow{\$} \mathcal{K}$
- $\text{Enc}_k(m) = k \oplus m$
- $\text{Dec}_k(c) = k \oplus c$
- correctness: $\text{Dec}_k(\text{Enc}_k(m)) = \text{Dec}_k(k \oplus m) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = m$

3 Perfect Secrecy

Intuitively, we want the ciphertext to leak no information to the adversary (except length). We need to define information and to capture this notion using probabilities so we can reason about it mathematically.

3.1 Probability Distribution

Notation Let X be a random variable taking values in the set $\mathcal{X}$, then $\Pr[X = x]$ denotes the probability that $x \in \mathcal{X}$ is chosen.

Distribution over key space $\mathcal{K}$ is defined by the output of Gen and usually *uniform*.

Distribution over message space $\mathcal{M}$ is usually not uniform, i.e., not all messages appear with the same probability. Distributions over $\mathcal{K}$ and $\mathcal{M}$ are usually independent of each other.

Distribution over ciphertext space $\mathcal{C}$ is defined by $\text{Enc}_k(m)$ and determined by distributions over $\mathcal{K}$ and $\mathcal{M}$.

3.2 Perfect Secrecy (continued)

Definition
A private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has *perfect secrecy* if for every probability distribution over $\mathcal{M}$, every $m \in \mathcal{M}$, and every $c \in \mathcal{C}$ with $\Pr[C = c] > 0$, it holds that

$$\Pr[M = m \mid C = c] = \Pr[M = m]$$

where $k \xleftarrow{\$} \mathcal{K}$ via Gen.

Given ciphertext c, an adversary does not gain any information about which message m was encrypted (with or without seeing c, the probabilities remain the same). This rules out ciphertext-only attacks such as those possible against classical ciphers.

Equivalently, when comparing messages from the same length class (or when the scheme leaks only length), for all $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$ and every $c \in \mathcal{C}$ it holds that

$$\Pr[\text{Enc}_k(m_0) = c] = \Pr[\text{Enc}_k(m_1) = c]$$

Which states that given ciphertext c, an adversary cannot tell if the corresponding plaintext is m_0 or m_1.

Proof.

$$\Pr[M = m \mid C = c] = \frac{\Pr[C = c \mid M = m] \Pr[M = m]}{\Pr[C = c]}$$

Bayes theorem;

$$\Pr[M = m] = \frac{\Pr[C = c \mid M = m] \Pr[M = m]}{\Pr[C = c]}$$

$$\forall m, \forall c \text{ with } \Pr[C = c] > 0, \quad \Pr[C = c \mid M = m] = \Pr[C = c]$$

$$\forall m_0, m_1 \in \mathcal{M}, \quad \Pr[C = c \mid M = m_0] = \Pr[C = c] = \Pr[C = c \mid M = m_1]$$

□

4 Security Games

A perfectly secure encryption scheme gives the adversary no information, therefore the adversary should have no advantage over an algorithm that outputs a random guess ($\frac{1}{2}$). If the definition must hold for any pair of messages, we let the adversary pick two messages of the same length. The challenger encrypts one of the messages at random and the adversary must guess which message was encrypted.

4.1 Perfect Indistinguishability Security Game

Definition
Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ and let $\mathcal{A}$ be an adversary (a stateful algorithm). We define the security game for perfect indistinguishability in the presence of an eavesdropper $\mathsf{Game}^{\text{EAV}}_{\mathcal{A},\Pi}$ between $\mathcal{A}$ and a challenger as follows:

$\mathsf{Game}^{\text{EAV}}_{\mathcal{A},\Pi}$:
$(m_0, m_1) \leftarrow \mathcal{A}$ //$\mathcal{A}$ picks a pair of messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$
$k \xleftarrow{\$} \text{Gen}(\cdot)$ //Challenger picks a random key $k \in \mathcal{K}$ running Gen
$b \xleftarrow{\$} \{0,1\}$ //Challenger picks a bit $b \in \{0,1\}$ uniformly at random
$c \leftarrow \text{Enc}_k(m_b)$ //Challenger computes a ciphertext c using k and m_b
$b' \leftarrow \mathcal{A}(c)$ //After receiving c, $\mathcal{A}$ guesses which message was encrypted
return $b = b'$ //$\mathcal{A}$ wins if $b = b'$ is 1 (true)

Π has perfect indistinguishability if for every adversary $\mathcal{A}$:

$$\Pr[\mathsf{Game}^{\text{EAV}}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$$

4.2 Perfect Secrecy of One-Time Pad

For One-Time Pad encryption, observe that for arbitrary $m \in \mathcal{M}$ and $c \in \mathcal{C}$, there is exactly one key $k \in \mathcal{K}$ such that $\text{Enc}_k(m) = k \oplus m = c$. Therefore

$$\Pr[\text{Enc}_k(m) = c] = \Pr[k \oplus m = c] = \Pr[k = m \oplus c] = \frac{1}{2}^{\lambda}$$

Limitations of OTP

- Every encryption must be done with a new key k. If a key is reused, then $c_1 \oplus c_2 = m_1 \oplus m_2$; if one plaintext is known, the other is recovered as $m_2 = m_1 \oplus c_1 \oplus c_2$.
- k is at least as long as message m. This is unusable in practice and key management will be a nightmare.

OTP with Shorter Keys?

Proof.

(by contrapositive) Assume that $|\mathcal{K}| < |\mathcal{M}|$. Fixing $m_0 \in \mathcal{M}$ and $k_0 \in \mathcal{K}$, we get $c_0 := \text{Enc}_{k_0}(m_0)$ and we have $\Pr[\text{Enc}_k(m_0) = c_0] > 0$. Let $S := \{\text{Dec}_k(c_0) : k \in \mathcal{K}\}$ be the messages to which c_0 decrypts for every key $k \in \mathcal{K}$.

Since $|S| \leq |\mathcal{K}| < |\mathcal{M}|$, we can choose $m_1 \in \mathcal{M} \backslash S$. If a key $k_1 \in \mathcal{K}$ existed for which $\text{Enc}_{k_1}(m_1) = c_0$, then the correctness of Π implies $\text{Dec}_{k_1}(c_0) = \text{Dec}_{k_1}(\text{Enc}_{k_1}(m_1)) = m_1$, which implies $m_1 \in S$, contrary to our choice of $m_1 \in \mathcal{M} \backslash S$. So no key exists which means $\Pr[\text{Enc}_k(m_1) = c_0] = 0$.

As $\Pr[\text{Enc}_k(m_0) = c_0] > 0$ and $\Pr[\text{Enc}_k(m_1) = c_0] = 0$, Π does not have perfect secrecy, contrary to what we are given, therefore our initial assumption is wrong and $|\mathcal{K}| \geq |\mathcal{M}|$. □

Shannon's Theorem

Theorem Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ with $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$. Then Π is perfectly secret if and only if:

- Every key $k \in \mathcal{K}$ is chosen with equal probability $\frac{1}{|\mathcal{K}|}|$ by Gen
- For every $m \in \mathcal{M}$ and every $c \in \mathcal{C}$, there exists a unique key $k \in \mathcal{K}$ such that $c = \text{Enc}_k(m)$

Characteristically, this means all perfectly secret encryption schemes will have key $k \in \mathcal{K}$ chosen uniformly at random, ciphertext $c \in \mathcal{C}$ can encrypt every message $m \in \mathcal{M}$ with the same probability. Every c should also have the same (non-zero) probability to occur, and the length of the key must be at least the length of the message.

5 Modelling Security of Private-Key Encryption

Relative to information-theoretic security, computational security has two main relaxations; 1) adversary $\mathcal{A}$ has a limited maximum runtime, 2) $\mathcal{A}$ can break security with some very small probability ϵ.

5.1 Probabilistic Polynomial Time Algorithms

Definition
An algorithm A is called efficient if there exists a constant $c > 0$ such that its running time on inputs of length n is $O(n^c)$.

5.2 Negligible Functions

Definition
A function $\epsilon : \mathbb{N} \to [0, 1]$ is called negligible if for any $c > 1$ there exists an N such that for all $n > N$ it holds that $\epsilon(n) < \frac{1}{n^c}$. We denote the class of negligible functions as $\text{negl}(n)$.

5.3 Asymptotic Security

Definition
A cryptographic scheme Π is (t, ϵ)-secure relative to security parameter λ if any randomized adversary $\mathcal{A}$ running polynomial time $t(\lambda)$ succeeds in breaking Π with at most negligible probability $\epsilon(\lambda)$.

5.4 Computational Indistinguishability: EAV-Security

Definition
A private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has indistinguishable encryption in the presence of an eavesdropper or is EAV-secure if for every PPT adversary $\mathcal{A}$ there is a negligible function negl such that for all λ,

$$\Pr[\mathsf{Game}^{\text{EAV}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda)$$

Pseudorandom Generators

These are generators which are computational relaxations of true randomness. A PRG is deterministic once its seed is fixed, but without the seed its output is computationally indistinguishable from uniform to efficient adversaries; the output distribution itself is not uniform over the longer output space.
Definition
Let l be a polynomial and let $G : \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)}$, $s \mapsto G(s)$ be a deterministic polynomial-time algorithm. We say G is a pseudorandom generator (PRG) if the following holds:

- Expansion: For every λ it holds that $l(\lambda) > \lambda$. We call l the expansion factor of G.
- Pseudorandomness: For any PPT algorithm D, there is a negligible function negl such that

$$|\Pr[D(G(s)) = 1] - \Pr[D(r) = 1]| \leq \text{negl}(\lambda)$$

  where $s \xleftarrow{\$} \{0,1\}^\lambda$ and $r \xleftarrow{\$} \{0,1\}^{l(\lambda)}$.

Do such generators actually exist? Theory proves that PRGs can be constructed from One-Way Functions (OWF). The existence of OWFs is not proven either, but we have strong indicators from practice. Henceforth we assume OWFs exist and explore how to build PRGs from them.

6 One-Way Functions (OWFs) and Permutations

Definition
A function $f : \{0,1\}^* \to \{0,1\}^*$ is an OWF if the following hold:

- Efficiency: There exists a PPT algorithm M_f such that $M_f(x) = f(x)$ for all x.
- Hard to invert: For every PPT algorithm $\mathcal{A}$ there is a negligible function negl such that for all λ

$$\Pr[\mathsf{Game}^{\text{INV}}_{\mathcal{A},f}(\lambda) = 1] \leq \text{negl}(\lambda)$$

Definition
Let $f : \{0,1\}^* \to \{0,1\}^*$ be an OWF. We say that f is a one-way permutation (OWP) if it is injective and length-preserving (i.e., $|f(x)| = |x|$)

7 Hard-Core Predicates

One-Wayness vs Secrecy

- given one-way function f and its output $y = f(x)$, does y reveal information about x?
- Yes, it's possible y leaks information about x even though f is hard to invert.
- e.g., let g be an OWF, then $f(x_1, x_2) \stackrel{\text{def}}{=} (x_1, g(x_2))$ is hard to invert but reveals x_1

Definition
A function $hc : \{0,1\}^* \to \{0,1\}$ is a hard-core (HC) predicate of an OWF f if the following hold:

- Efficiency: hc can be computed in polynomial time.

- Secrecy: For every PPT algorithm $\mathcal{A}$ there is a negligible function negl such that

$$\Pr_{x \xleftarrow{\$} \{0,1\}^\lambda}[\mathcal{A}(1^\lambda, f(x)) = hc(x)] \leq \frac{1}{2} + \text{negl}(\lambda)$$

basically $hc(x)$ should be easy to compute knowing x but hard to compute knowing only $f(x)$.

7.1 HC Predicate for any OWF

For a given OWF f, we can construct another OWF g and an HC predicate hc for g in a general way.
Goldreich-Levin HC Predicate (1989) Let f be an OWF/OWP. Construct g and hc as

- $g(x, r) \stackrel{\text{def}}{=} (f(x), r)$ for some random bit string $r \xrightarrow{\$} \{0,1\}^\lambda$ of length $|r| = |x|$
- $hc(x, r) \stackrel{\text{def}}{=} \bigoplus_{i=1}^{\lambda}(x_i r_i)$ is the inner product of x and r modulo 2.

The Goldreich–Levin theorem shows that for any OWF/P f, the pair $(f(x), r)$ hides the inner product of x with a random string r.

7.2 Building PRGs from OWF with HC

Theorem
Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be an OWP and let $hc : \{0,1\}^\lambda \to \{0,1\}$ be an HC predicate of f. Then

$$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}, s \mapsto f(s) \| hc(s)$$

is a PRG with (minimal) expansion factor $l(\lambda) = \lambda + 1$

Intuitively (as an informal proof), G is a PRG, and because s is truly random, then so is $f(s)$, and $hc(s)$ 'looks like' a random bit even if $f(s)$ is known.

For real applications, we need arbitrary expansion. Thus a PRG $\hat{G}$ with $l(\lambda) = \lambda + \text{poly}(\lambda)$ and input $s \in \{0,1\}^\lambda$ can be created as follows:

Algorithm 1 Iterative PRG Construction

```
t_0 := s
for i = 0, ..., poly(λ) − 1 do
    s_i := t_i[0 : λ]          ▷ Set s_i to the first λ bits of t_i
    r_i := t_i[λ : λ + i]      ▷ Set r_i to the accumulated i output bits
    t_{i+1} := G(s_i) ‖ r_i
end for
return t_{poly(λ)}
```

Private-Key Encryption

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 13-17 Oct

Overview

Building practical private-key encryption schemes means they need to be computationally secure with reusable fixed-length keys.

This lecture covers indistinguishability under chosen plaintext attacks (CPA), PRFs, private-key encryption from PRFs, block ciphers and modes of operation, block cipher constructions and stream cipher constructions.

8 Indistinguishability for Multiple Encryptions

Given a private-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, For $\mathsf{Game}^{\text{EAV-MULT}}_{\mathcal{A},\Pi}(\lambda)$, we have a PPT adversary $\mathcal{A}$ who outputs two messages m_0 and m_1. $\mathcal{A}$ can learn the encryption of any plaintext message by having the same message at the same position in both vectors of its output. This attack is stronger than basic known-plaintext attacks (KPA), as the adversary controls the plaintexts for which it knows the ciphertext.

The adversary's output is chosen before $\mathcal{A}$ can observe any ciphertexts. Once $\mathcal{A}$ receives c, it can no longer ask for encryptions of further messages. This is a non-adaptive attack as the adversary cannot use knowledge of plaintext-ciphertext pairs to choose its output or other plaintexts for which to learn the ciphertext.

In practice, Eve ($\mathcal{A}$) may send Alice some message m_i to forward to Bob over the secure channel as ciphertext c_i. Eve can then observe the relationship between m_i and c_i. This may happen multiple times, and after each time, Eve may use any information it learns to adapt its strategy. However, we want indistinguishability to hold for unknown ciphertexts even if $\mathcal{A}$ obtains ciphertexts c_i for any message m_i of its choice at any time during an attack.

8.1 Oracles

We want to give Eve the ability to encrypt any message of her choice with key k. Eve knows the specification of $\text{Enc}(\cdot)$[1] but needs k to encrypt. Giving Eve k trivializes the indistinguishability game, since she can just decrypt the challenge message immediately.

So Eve is given *oracle access* to $\text{Enc}_k(\cdot)$, which hides the key but allows Eve to query it with any message of her choice m_i and receive $c_i := \text{Enc}_k(m_i)$. From Eve's perspective, oracles are efficient black boxes that always produce a correct output. A PPT adversary may adaptively interact with an oracle a polynomial number of times.

8.2 Chosen-Plaintext Attack Indistinguishability Security Game

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, let λ denote a security parameter and let $\mathcal{A}$ be a PPT adversary. We define the security game for an IND-CPA $\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda)$ between $\mathcal{A}$ and a challenger as follows:

$\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda)$:

$k \xleftarrow{\$} \text{Gen}(1^\lambda)$

$(m_0, m_1) \leftarrow \mathcal{A}^{\text{Enc}_k(\cdot)}(1^\lambda)$ `//adversary chooses` $m_0, m_1 \in \mathcal{M}$ `with` $|m_0| = |m_1|$

$b \xleftarrow{\$} \{0,1\}$

$c \leftarrow \text{Enc}_k(m_b)$

$b' \leftarrow \mathcal{A}^{\text{Enc}_k(\cdot)}(c)$

[1] Because of Kerckhoffs' principle.

return $b = b'$

Π is IND-CPA if for every PPT adversary $\mathcal{A}$ there is a negligible function negl such that for all λ

$$\Pr[\mathsf{Game}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$$

where the probability is over the randomness of $\mathcal{A}$ and the randomness in the experiment.

9 Pseudorandom Functions

Recall that PRGs, given a random input, output bit strings that are indistinguishable from random. Can we have a function with pseudorandom output when given non-random input? Yes, as the function itself is chosen at random.

Definition

- A function $F : \{0,1\}^{l_{\mathrm{in}}} \to \{0,1\}^{l_{\mathrm{out}}}$ is called a pseudorandom function if its outputs remain computationally indistinguishable from a chosen random function $f : \{0,1\}^{l_{\mathrm{in}}} \to \{0,1\}^{l_{\mathrm{out}}}$.
- A keyed function $F : \{0,1\}^{l_{\mathrm{key}}} \times \{0,1\}^{l_{\mathrm{in}}} \to \{0,1\}^{l_{\mathrm{out}}}$ has two inputs: a key k which uniquely identifies F in the family of functions, and some variable x.

Fixing a keyed function F means choosing $k \in \{0,1\}^{l_{\mathrm{key}}}$ and computing outputs on input $x \in \{0,1\}^{l_{\mathrm{in}}}$ as $F_k(x) \stackrel{\mathrm{def}}{=} F(k,x)$. Therefore picking a function at random is equivalent to picking a key at random and fixing a function. Since $k \in \{0,1\}^{l_{\mathrm{key}}}$, $2^{l_{\mathrm{key}}}$ unique functions $F_k : \{0,1\}^{l_{\mathrm{in}}} \to \{0,1\}^{l_{\mathrm{out}}}$ can be fixed. We will set $l_{\mathrm{key}} = l_{\mathrm{in}} = l_{\mathrm{out}} = \lambda$ for simplicity, but not necessary for definition.

9.1 Choosing a Random Function f

A random function $f : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ can be described as a unique look-up table. The number of different random functions $f : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ is equal to the number of different $\lambda \cdot 2^{\lambda}$-bit tables, namely $2^{\lambda \cdot 2^{\lambda}}$.

Since the description of one such table is exponential in λ (namely $\lambda \cdot 2^{\lambda}$ bits), no PPT algorithm $\mathcal{D}$ may store an explicit random table for f. Therefore, instead of giving $\mathcal{D}$ a description of f, $\mathcal{D}$ is given oracle access to f. To get $f(x)$, $\mathcal{D}$ needs to send x to its oracle $f(\cdot)$, and receive $f(x)$. Since $\mathcal{D}$ is PPT, the number of queries remains polynomial in λ.

9.2 Defining and Building Pseudorandom Functions

Definition

Let $F : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an efficient keyed function. F is a PRF if for all PPT algorithms $\mathcal{D}$, there exists a negligible function ϵ such that

$$|\Pr[\mathcal{D}^{F_k(\cdot)}(\lambda) = 1] - \Pr[\mathcal{D}^{f(\cdot)}(\lambda) = 1]| \leq \epsilon(\lambda)$$

where the key $k \stackrel{\$}{\leftarrow} \{0,1\}^{\lambda}$ and $f : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ is a randomly chosen function.

$\mathcal{D}$ should distinguish whether the oracle given is $F_k(\cdot)$ or $f(\cdot)$ with at most negligible probability, and by necessity, on input x, $\mathcal{D}$ should not be able to distinguish whether the oracle computed $F_k(x)$ or $f(x)$. PRF security implies that recovering the secret key from oracle access should be infeasible, but "inversion" is not the defining property of a PRF.

From 7.2, we can build PRGs from OWFs. Similarly we can build PRFs from PRGs. Given a PRG G with $l(\lambda) = 2\lambda$, we define $G(s) := G_0(s) \| G_1(s)$ where $G_0(s)$ and $G_1(s)$ denote the left and right λ bits of $G(s)$, respectively. We define $F_k : \{0,1\} \to \{0,1\}^{\lambda}$ with $F_k(0) := G_0(k)$ and $F_k(1) := G_1(k)$. Then $F(k)$ is pseudorandom since 0 and 1 are mapped to pseudorandom strings. Next, $F_k : \{0,1\}^2 \to \{0,1\}^{\lambda}$ with $F_k(ij) := G_j(G_i(k))$ and $i, j \in \{0,1\}$.

Goldreich-Goldwasser-Micali utilises G, G_0, G_1 as above. For every $k \in \{0,1\}^{\lambda}$ define

$$F_k : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}, F_k(x_1 \ldots x_{\lambda}) := G_{x_{\lambda}} \circ \ldots \circ G_{x_1}(k)$$

where x_i is the i-th bit input x and $\circ$ denotes function composition.

10 Private-Key Encryption from PRFs

We know that deterministic OTP with PRG is not IND-CPA. To achieve the latter we need to randomize the construction and replace the PRG with a PRF.

Definition
Let $\{F_k : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{k\in\{0,1\}^\lambda}$ be a PRF family and $\mathcal{M} \stackrel{\text{def}}{=} \{0,1\}^\lambda$. A OTP with PRF F is specified as follows:

- $\text{Gen}(\lambda)$: return $k \xrightarrow{\$} \{0,1\}^\lambda$
- $\text{Enc}_k(m)$: pick $r \xrightarrow{\$} \{0,1\}^\lambda$, return $c := (c_1, c_2) = (r, F_k(r) \oplus m)$
- $\text{Dec}_k(c)$: return $m := F_k(c_1) \oplus c_2$

Theorem
If $F_k : 0, 1^\lambda \to \{0,1\}^\lambda$ is a PRF, then OTP with PRF is IND-CPA.

Proof.

Oracle $\mathcal{O}$: if contains f, then $\mathcal{O}(r) = f(r)$. If it contains F_k, then $\mathcal{O}(r) = F_k(r)$.

The goal of D: Determine if oracle $\mathcal{O}$ contains F_k or f.

D needs to simulate IND-CPA game with $\mathcal{A}$:

- The encryption oracle $\text{Enc}_k(\cdot)$: pick $r \xleftarrow{\$} \{0,1\}^\lambda$ and return $c := (r, \mathcal{O}(r) \oplus m_b)$.
- The challenger: after receiving (m_0, m_1), pick $b \xleftarrow{\$} \{0,1\}$, pick $r \xleftarrow{\$} \{0,1\}^\lambda$, and return $c := (r, \mathcal{O}(r) \oplus m_b)$.

Since $\mathcal{A}$ runs in polynomial time, so does D. □

10.1 Analysis of Reduction

PRF security is defined as;

$$|\Pr[D^{F_k(\cdot)}(\lambda) = 1] - \Pr[D^{f(\cdot)}(\lambda) = 1]| \leq \text{negl}(\lambda)$$

IND-CPA security is defined as;

$$|\Pr[\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}| \leq \text{negl}(\lambda)$$

D must output 1 if $\mathcal{A}$ wins the IND-CPA game. When $\mathcal{O}$ contains PRF F_k, $\mathcal{A}$ is playing the game against OTP with PRF. Meaning $\Pr[D^{F_k(\cdot)}(\lambda) = 1] = \Pr[\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1]$. When $\mathcal{O}$ contains random function f, $\mathcal{A}$ is playing against OTP with f replacing the PRF,$\mathcal{A}$'s advantage is in if the oracle encryption uses the same randomness as the ciphertext c. $\Pr[D^{f(\cdot)}(\lambda) = 1] \leq \frac{1}{2} + \Pr[r_{m_b} = r_{q_i} \in \text{Enc}_k(\cdot)] \leq \frac{1}{2} + \frac{q}{2^\lambda}$. where the same r is used to encrypt m_b and some query q_i to $\text{Enc}_k(\cdot)$, and q is the number of queries by $\mathcal{A}$ to $\text{Enc}_k(\cdot)$.

Proof.

We have that $\Pr[D^{F_k(\cdot)}(\lambda) = 1] = \Pr[\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1]$, and $\Pr[D^{f(\cdot)}(\lambda) = 1] \leq \frac{1}{2} + \frac{q}{2^\lambda}$.

Then, subtracting both terms yields

$$\Pr[D^{F_k(\cdot)}(\lambda) = 1] - \Pr[D^{f(\cdot)}(\lambda) = 1] \geq \Pr[\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2} - \frac{q}{2^\lambda}$$

Thus, $\forall$ PPT $\mathcal{A}$, we have that;

$$\Pr[\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \frac{q}{2^\lambda} + \text{negl}(\lambda)$$

Since $\mathcal{A}$ is PPT, q is polynomial and therefore $\frac{q}{2^\lambda}$ is negligible;

$$\Pr[\mathsf{Game}^{\text{CPA}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \frac{1}{2} + \text{negl'}(\lambda)$$

Therefore, OTP with PRF is IND-CPA secure if $F_k(\cdot)$ is a PRF. □

When message m is of variable-length;

- Divide m into λ-bit blocks $m_1, \ldots, m_n$ and process each m_i separately.
- If $x = |m_n| < \lambda$, then pad as $m_n \| 10^{\lambda - x - 1}$.

the OTP with PRF construction is then;

- $\text{Gen}(\lambda)$: return $k \xrightarrow{\$} \{0,1\}^\lambda$
- $\text{Enc}_k(m)$: for each m_i return $c_i := (c_{i1}, c_{i2}) = (r_i, F_k(r_i) \oplus m_i)$ with $r \xrightarrow{\$} \{0,1\}^\lambda$
- $\text{Dec}_k(c)$: for each c_i return $m_i := F_k(c_i 1) \oplus c_i 2$

This construction is **not** very practical, however, because $|c| = n \cdot 2\lambda$ bits, i.e., the ciphertext is twice as long as the message. Some key things to note; IND-CPA is the baseline security for any encryption scheme, and is achieved through randomization. Typically IND-CPA is proven by reduction onto a computationally hard problem (e.g., inversion of OWFs, factoring). So how can we build IND-CPA private key encryption schemes which are practical?

11 Block Ciphers & Modes of Operation

11.1 Pseudorandom Permutations (PRPs)

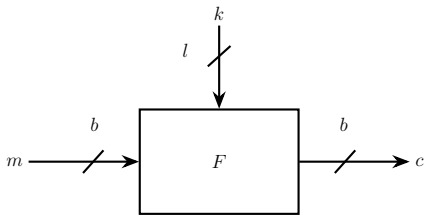
Definition
Let $F : \{0,1\}^\lambda \times \{0,1\}^b \to \{0,1\}^b$ be an efficient keyed permutation family for fixed key length λ and block length b. F is a pseudorandom permutation (PRP) if for all PPT algorithms, there exists a negligible function negl, such that

$$|\Pr[D^{F_k, F_k^{-1}}(\lambda) = 1] - \Pr[D^{f, f^{-1}}(\lambda) = 1]| \leq \text{negl}(\lambda)$$

where $k \xleftarrow{\$} \{0,1\}^\lambda$ is a random key and f is a random permutation over $\{0,1\}^b$.

F is a keyed permutation if for any k the function F_k, is one-to-one. F_k is length-preserving and one-to-one and thus can be inverted, i.e., there exists $F_k^{-1}(F_k(m)) = m$.

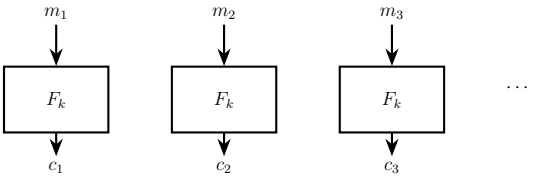
11.2 Block Ciphers



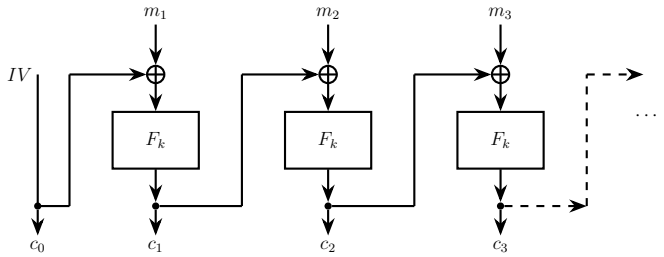
Definition
A block cipher is an efficient, keyed permutation $F : \{0,1\}^\lambda \times \{0,1\}^b \to \{0,1\}^b$, where the security parameter λ determines the key length and parameter b a fixed block size.

Block ciphers are PRPs; their output must be indistinguishable from a random permutation. Modern block ciphers use F_k as encryption and F_k^{-1} as decryption. They can only encrypt a single block of size b bits, and to build practical encryption from these block ciphers, **modes of operations** are used.

Electronic Code Book (ECB) is a mode of operation which encrypts $m = (m_1, \ldots, m_n)$ by computing $c_i = F_k(m_i)$ for each $1 \leq i \leq n$. ECB is deterministic and thus not IND-EAV or IND-CPA secure, and thus should never be used.

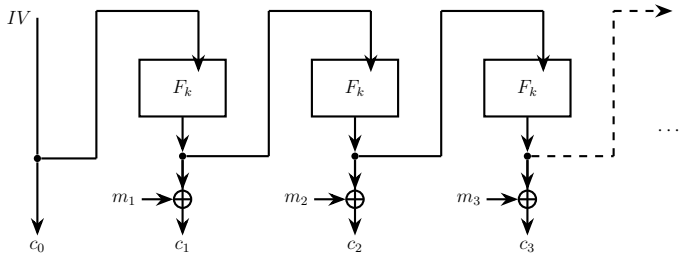


Cipher Block Chaining (CBC) Let $\text{IV} \xleftarrow{\$} \{0,1\}^b$ be an initialisation vector. CBC mode encrypts a message $m = (m_1, \ldots, m_n)$ by computing $c_i = F_k(m_i \oplus c_{i-1})$ for each $1 \leq i \leq n$ where $c_0 = \text{IV}$. Decryption is specified as $m_i = F_k^{-1}(c_i) \oplus c_{i-1}$, and is parallelisable. CBC is IND-CPA secure if F is a PRP and IV is randomly chosen for every new message.



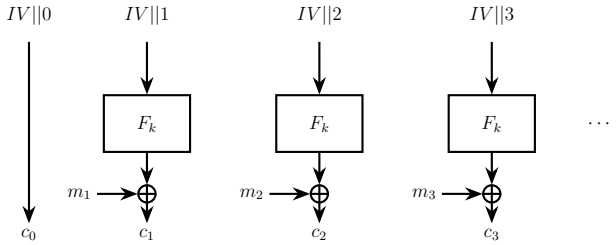
Output Feedback (OFB) Let $\text{IV} \xleftarrow{\$} \{0,1\}^b$ be an initialisation vector. The OFB mode encrypts a message $m = (m_1, \ldots, m_n)$ by computing $c_i = F_k^i(\text{IV}) \oplus m_i$ for each $1 \leq i \leq n$. Analogously, decryption is specified as $m_i = F_k^i(IV) \oplus c_i$.

- Enc and Dec are not parallelizable but $F_k^i(\text{IV})$ can be precomputed for all i.
- Enc and Dec only require F_k, which can be a PRF.
- OFB is IND-CPA if F is a PRF and a new IV is chosen randomly for every new message.



Counter (CTR) Let $\text{IV} \xleftarrow{\$} \{0,1\}^b$ be an initialisation vector. The CTR mode encrypts a message $m = (m_1, \ldots, m_n)$ by computing $c_i = F_k(\text{IV} + i) \oplus m_i$ for each $1 \leq i \leq n$. Analogously, decryption is specified as $m_i = F_k(\text{IV} + i) \oplus c_i$.

- Blocks are processed independently making Enc and Dec parallelizable.
- Enc and Dec only require F_k, which can be a PRF.
- CTR is IND-CPA if F is a PRF and the counter inputs $\text{IV} + i$ never repeat under the same key; in practice, the IV/nonce must be unique with negligible collision probability and counters must not wrap around.



12 Block Cipher Constructions

Practical approaches to building block ciphers include Feistel networks (in DES and 3DES), and substitution-permutation networks (e.g., AES).

12.1 Feistel Networks

The goal of Feistel networks is to construct an invertible function from a non-invertible one. In round i update, a b-bit sized state (L_i, R_i) using a (non-invertible) function f_i:

$$L_{i+1} := R_i \text{ and } R_{i+1} := L_i \oplus f_i(R_i)$$

f_i is keyed with a round key k_i that is derived from k. And each round i is invertible even if f_i is not:

$$R_i := L_{i+1} \text{ and } L_i := R_{i+1} \oplus f_i(L_{i+1})$$

Any Feistel network is a permutation, but pseudorandomness requires assumptions on the round functions and sufficiently many rounds. For example, Luby–Rackoff-type results use pseudorandom round functions: three rounds give a weak PRP and four rounds give a strong PRP under standard assumptions.

For example, a 3-round Feistel network we get:

- $\text{Gen}(\lambda)$: return a key $k = (k_0, k_1, k_2)$ for $k_0 \xleftarrow{\$} \{0,1\}^\lambda, k_1 \xleftarrow{\$} \{0,1\}^\lambda, k_2 \xleftarrow{\$} \{0,1\}^\lambda$
- $\text{Enc}_k(\cdot)$ and $\text{Dec}_k(\cdot)$ use $f_i \stackrel{\text{def}}{=} F_{k_i}$ for $0 \le i \le 2$.

Data Encryption Standard (DES) 56-bit key k, 64-bit blocks, 16-round Feistel network. After the first practical attacks on DES in 1998, the proposed fix was to increase the key length.

Let $F : \{0,1\}^\lambda \times \{0,1\}^b \to \{0,1\}^b$ be a block cipher, and then define a new block cipher $F' : \{0,1\}^{2\lambda} \times \{0,1\}^b \to \{0,1\}^b$ as $F'_k(m) \stackrel{\text{def}}{=} F_{k_2}(F_{k_1}(m))$ with $k \xleftarrow{\$} \{0,1\}^{2\lambda}$ i.e., $k = (k_1, k_2)$.

The issue then arises as a meet-in-the-middle attack. If $c = F'_k(m)$, then $F_{k_1}(m) = F^{-1}_{k_2}(c)$. this can be done with $2 \cdot 2^\lambda$ operations $\to O(2^\lambda)$ complexity. which is similar to single encryption.

Triple Encryption defines a new block cipher $F'\{0,1\}^{3\lambda} \times \{0,1\}^b \to \{0,1\}^b$ as $F'_k(m) \stackrel{\text{def}}{=} F_{k_3}(F_{k_2}(F_{k_1}(m)))$, with $k \xleftarrow{\$} \{0,1\}^{3\lambda}$ i.e., $k = (k_1, k_2, k_3)$. Brute force attack complexity in $O(2^{3\lambda})$, while MITM is still applicable but requires $O(2^{2\lambda})$.

Backwards-Compatible Triple Encryption Define a new block cipher $F' : \{0,1\}^{2\lambda} \times \{0,1\}^b \to \{0,1\}^b$ as $F'_k(m) \stackrel{\text{def}}{=} F_{k_1}(F^{-1}_{k_2}(F_{k_1}(m)))$ with $k \xleftarrow{\$} \{0,1\}^{2\lambda}$ i.e., $k = (k_1, k_2)$. Brute force attack complexity in $O(2^{2\lambda})$, there is a CPA running in $O(2^\lambda)$ requiring 2λ plaintexts. Backwards compatibility means that if $k_1 = k_2$, then $F'_k(m) = F_{k_1}(m)$ as in single encryption.

12.2 Confusion-Diffusion Paradigm

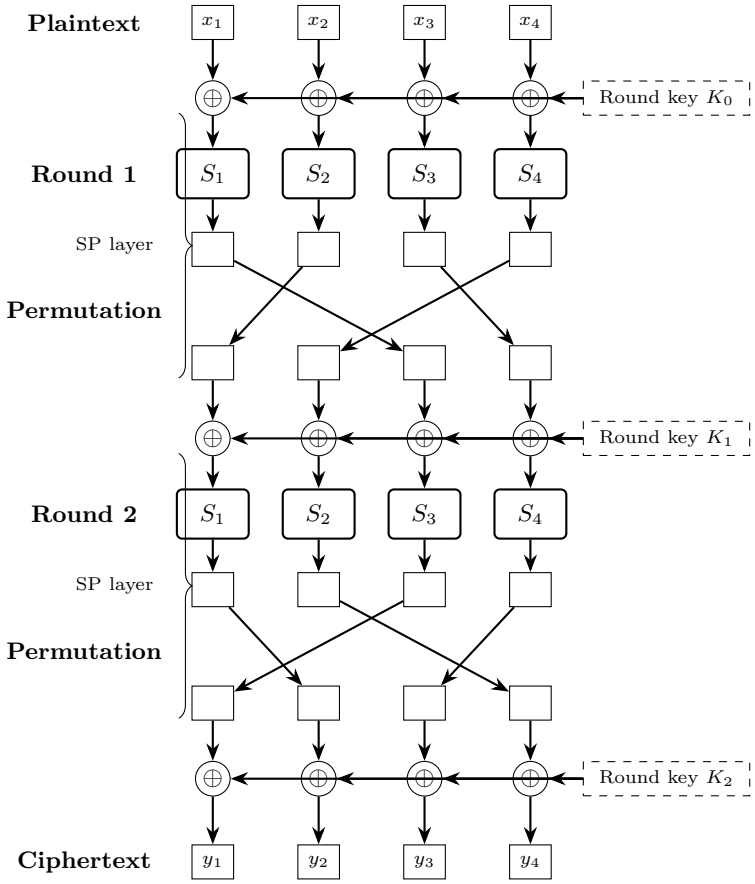
Introduced by Shannon as a general design approach for encryption schemes in his 1945 report 'A Mathematical Theory of Cryptography'.

- **Confusion**: Each ciphertext bit should depend on multiple key bits. The goal is to hide the relationship between key and ciphertext.
- **Diffusion**: Flipping one bit in the plaintext flips approximately half of the bits in the ciphertext. The goal is to hide the relationship between plaintext and ciphertext.

Goal: every bit of the output should depend on every bit of the input in a non-trivial manner (avalanche effect).

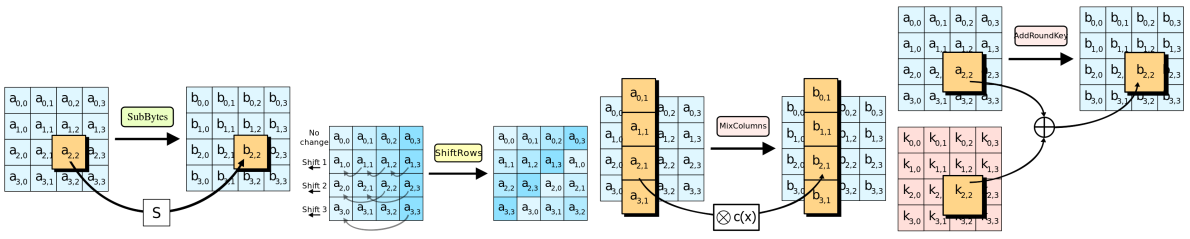
12.3 Substitution-Permutation (SPN) Networks

SPNs are one approach to achieve confusion and diffusion.



- **Substitution**: Provides confusion, implemented via (non-linear) S-boxes (look-up tables).
- **Permutation**: Provides diffusion, implemented via (linear) P-boxes.
- **Round**: Combination of an S-box layer, a P-box layer, and the XORing of a round key k_i for $0 \leq i \leq r-1$. P-box is usually omitted for $i = r-1$.

Advanced Encryption Standard (AES) is a version of the Rijndael cipher which won the NIST block cipher competition, and standardized as AES by NIST in 2001. It has three different key lengths (128, 192, 256 bit), a block length of 128 bit, between 10, 12 and 14 SPN rounds depending on the key length, and its state is defined by a 4×4-byte matrix (filled column-wise). Encryption begins with an initial `AddRoundKey`; the main rounds apply `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey`; the final round omits `MixColumns`.



(a) SubBytes (b) ShiftRows (c) MixColumns (d) AddRoundKey

Figure 1: AES Round Functions

AES as a block cipher is still considered secure for its standard key sizes, but a mode of operation can still be insecure or misused; for example, ECB leaks patterns and CTR/GCM fail under nonce reuse. No known methods of differential and linear cryptanalysis have been found which are faster than brute-force.

'Biclique' attack (2011) "breaks" AES-128/-192/-256 with attack complexities of $2^{126.1}, 2^{189.7}, 2^{254.4}$, no real practical threat.

12.4 Stream Ciphers

Stream ciphers are highly efficient encryption mechanisms that process arbitrary-sized messages by default. They operate by extending a secret seed, or key k, into a pseudorandom keystream using a pseudorandom generator (PRG) G. The encryption process is computed via $c := G(k) \oplus m$, while decryption recovers the plaintext via $m := G(k) \oplus c$. While exceptionally fast in practice, reusing the same key for different messages creates severe security problems. To achieve Chosen-Plaintext Attack (CPA) security, the initial seed must be extended with a non-secret initialization vector (IV), forming the seed (k, IV).

RC4 Cipher

Designed by Ron Rivest in 1987, RC4 was historically adopted in protocols like WEP and TLS due to its simplicity and speed. It utilizes a variable-length key of 5 to 256 bytes to initialize a 256-byte internal state array S via a Key-Scheduling Algorithm (KSA). A Pseudorandom Generation Algorithm (PRGA) then continuously swaps state values to produce the keystream.

Despite its prior prevalence, RC4 is critically flawed and should not be used. The initial bytes of its keystream do not behave in a perfectly random manner. These statistical biases were exploited to break WEP in 2001, and advanced attack vectors compromised it within TLS in 2013. Consequently, major IT organizations—including the IETF, Microsoft, Google, and Mozilla—have universally banned RC4 from their standards.

ChaCha20

Introduced by Daniel Bernstein in 2008, ChaCha20 serves as a highly secure and extremely fast replacement for RC4. It operates on a 64-byte initial state matrix, structured as a 4×4 grid of 4-byte words. This initial state includes 16 bytes of constants, a 32-byte key, alongside an 8-byte counter and an 8-byte nonce.

The cipher's core mechanism is the Quarterround (QR) function, which updates four state elements in-place using Addition-Rotation-XOR (ARX) operations. This function updates each input word twice, leveraging addition as a non-linear operation to achieve fast diffusion, where a single flipped input bit affects an average of 12.5 output bits.

The precise ARX operations within the Quarterround function $QR(a, b, c, d)$ are executed as follows:

- $a += b$; $d \hat{} = a$; $d \lll= 16$;
- $c += d$; $b \hat{} = c$; $b \lll= 12$;
- $a += b$; $d \hat{} = a$; $d \lll= 8$;
- $c += d$; $b \hat{} = c$; $b \lll= 7$;

ChaCha20 implements these transformations sequentially to the columns and diagonals of the state matrix through a Doubleround (DR) operation. The full ChaCha20 PRG applies ten Doublerounds to the initial state s, formulated as $G(s) := DR^{10}(s) \oplus s$. ChaCha20 uses a 256-bit key and is considered highly secure; brute-force key search would require testing 2^{256} keys, but this should not be read as a blanket claim of "256-bit security for every security notion. Furthermore, the best known differential and linear cryptanalysis methods only succeed against 6 or 7 of the 20 rounds, demanding massive computational complexities. Currently, no known attacks on the full-round ChaCha20 cipher that are faster than a brute-force search.

Hash Functions

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 20-24 Oct

Motivation

Information communicated should remain unchanged and detectable when present. We need a mechanism that can deterministically create a short "description" of arbitrarily long inputs. Storing this, we are then able to check if it matches the output of the mechanism.

13 Hash Function Principles

13.1 Hash Functions

A hash function is a pair of algorithms $\Sigma = (\text{Gen}, H)$ specified as;

- **Gen**: a PPT algorithm taking security parameter λ as input, outputting a key $s \in \{0,1\}^\lambda$.
- $\boldsymbol{H}$: is a deterministic polynomial-time function keyed by s and defined as

$$H^s : \{0,1\}^* \to \{0,1\}^{l(\lambda)}, m \mapsto H^s(m)$$

Hash functions essentially compresses an arbitrary sized input m efficiently into a short $l(\lambda)$-sized output $h = H^s(m)$ which is also called the digest or fingerprint. Typically hash tables achieve $O(1)$ look-up time. However, in the presence of an adversary, are insufficient for cryptographic use cases.

13.2 Compression Functions

A compression function is a pair of algorithms $\Sigma = (\text{Gen}, H)$ specified as;

- **Gen**: a PPT algorithm taking security parameter λ as input, outputting a key $s \in \{0,1\}^\lambda$.
- $\boldsymbol{H}$: is a deterministic polynomial-time function keyed by s and defined as

$$H^s : \{0,1\}^{l'(\lambda)} \to \{0,1\}^{l(\lambda)}, m \mapsto H^s(m)$$

with $l'(\lambda) > l(\lambda)$.

Compression functions are defined for fixed-length inputs, rather than arbitrary ones. Generally, they are easier to construct (e.g., from block ciphers), and can be transformed into hash functions that accept arbitrary length inputs.

13.3 Security of Hash Functions

Generally, only secure if they are hard to invert.

Preimage Resistance (PR1)

Definition
A hash function $\Sigma = (\text{Gen}, H)$ with key generation function Gen, and keyed function $H^s : \{0,1\}^* \to \{0,1\}^{l(\lambda)}$ for some key $s \leftarrow \text{Gen}(\lambda)$ is PR1 if for any PPT adversary $\mathcal{A}$:

$$\Pr[\mathsf{Game}^{\text{Pre}}_{\mathcal{A},H}(\lambda) = 1] \leq \text{negl}(\lambda)$$

$\mathsf{Game}^{\text{Pre}}_{\mathcal{A},H}(\lambda)$:
$s \xleftarrow{\$} \text{Gen}(1^\lambda)$
$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$

$h := H^s(x)$
$x' \leftarrow \mathcal{A}(1^\lambda, s, h)$
if $H^s(x') = h$ **then**
  return 1 // $\mathcal{A}$ found a preimage
end
return 0

PR1 hash functions are a type of one-way function (OWF). The output of the adversary, x', does not have to be x for it to win, only that it has the same fingerprint. The key s is public, a clear difference to PRFs (we use H^s to denote that s is public, compared to F_k where the key must be secret). The game (and subsequent ones) can be applied to compression functions by sampling $x \xleftarrow{\$} \{0,1\}^{l'(\lambda)}$.

Second-Preimage Resistance PR2

Definition
A hash function $\Sigma = (\text{Gen}, H)$ with key generation function Gen, and keyed function $H^s : \{0,1\}^* \to \{0,1\}^{l(\lambda)}$ for some key $s \leftarrow \text{Gen}(\lambda)$ is PR2 if for any PPT adversary $\mathcal{A}$:

$$\Pr[\mathsf{Game}^{\text{Sec}}_{\mathcal{A},H}(\lambda) = 1] \leq \text{negl}(\lambda)$$

$\mathsf{Game}^{\text{Sec}}_{\mathcal{A},H}(\lambda)$:
  $s \xleftarrow{\$} \text{Gen}(1^\lambda)$
  $x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
  $x' \leftarrow \mathcal{A}(1^\lambda, s, x)$
  if $x \neq x'$ and $H^s(x) = H^s(x')$ **then**
    return 1 // $\mathcal{A}$ found a second preimage
  end
  return 0

Given an input and its fingerprint, the adversary has to find a second preimage that produces the fingerprint. Also called the "weak collision resistance".

For arbitrary length inputs, if H is PR2, H is also PR1. For a compression function $H : \{0,1\}^{l'(\lambda)} \to \{0,1\}^{l(\lambda)}$, "PR2 $\implies$ PR1" depends on how compressed the input is, specifically, whether $\frac{1}{2^{l'(\lambda)-l(\lambda)}}$ is $\text{negl}(\lambda)$.

13.4 Collision Resistant Hash Functions (CRHF)

Definition
A hash function $\Sigma = (\text{Gen}, H)$ with key generation function Gen, and keyed function $H^s : \{0,1\}^* \to \{0,1\}^{l(\lambda)}$ for some key $s \leftarrow \text{Gen}(\lambda)$ is collision-resistant (CR) if for any PPT adversary $\mathcal{A}$:

$$\Pr[\mathsf{Game}^{\text{HashColl}}_{\mathcal{A},H}(\lambda) = 1] \leq \text{negl}(\lambda)$$

$\mathsf{Game}^{\text{HashColl}}_{\mathcal{A},H}(\lambda)$:
  $s \xleftarrow{\$} \text{Gen}(1^\lambda)$
  $(x, y) \leftarrow \mathcal{A}(1^\lambda, s)$
  if $x \neq y$ and $H^s(x) = H^s(y)$ **then**
    return 1 // $\mathcal{A}$ found a collision
  end
  return 0

CR is the strongest security definition for a cryptographic hash function. If H is CR, then H is PR2; which is a strictly weaker property. Also called "strong collision resistance", is sometimes called "collision free"; a slight misnomer as collisions will always exist, but should be hard to find.

13.5 Keyed vs. Unkeyed Hash Functions

In practice, hash functions are unkeyed and deterministic, with λ denoting the length of the output.

$$H : \{0,1\}^* \to \{0,1\}\lambda$$

The problem is that a PPT algorithm that outputs a collision for a fixed function always exists in theory, it just outputs known colliding inputs. No one knows what the. values x and y should be, but how could we mathematically formalise the statement "no one knows..."?

With keyed hash function, an algorithm needs to store a collision for every key $s \in \{0,1\}^\lambda$ which is impossible for a polynomially-bounded adversary, as it would require exponential memory to store 2^λ pairs.

Real-world cryptographic hash functions are collision resistant for all practical purposes since colliding pairs are unknown and computationally hard to find.

Reduction proofs that reduce the security of a scheme to the collision resistance of a hash function are valid for both keyed and unkeyed definitions. Keys will be omitted for hash functions for simplicity.

13.6 Finding Collisions

Assume $H : \{0,1\}^* \to \{0,1\}^l$ is a CRHF.
Brute-Force Search for Collisions

1. Compute $\boldsymbol{H} \stackrel{\text{def}}{=} \{H(x) | \forall x \in \{0,1\}^l\}$. If a collision is found, terminate.
2. If no collision found, $\forall h \in \{0,1\}^l$ we exactly one $x \in \{0,1\}^l$ such that $H(x) = h$.
3. Compute $H(x')$ for any fresh input $x' \notin \{0,1\}^l$, e.g., a string of length $l+1$.
4. We now have 2^{l+1} distinct inputs, but only 2^l distinct outputs.
   There must be at least one x, with $H(x) \in \boldsymbol{H}$ such that $H(x) = H(x')$.

This method takes exponential $O(2^l)$ time. is there a better algorithm?
Assume CRHF $H : \{0,1\}^* \to \{0,1\}^l$ acts like a random function.
Birthday Attacks

1. Choose $q \approx \sqrt{2^l}$ and arbitrary inputs $m_1, \ldots, m_q \in \{0,1\}^l$.
2. Compute $h_i = H(m_i) \forall 1 \leq i \leq q$.
3. There is a collision $h_i = h_j$ with constant probability; for $q \approx \sqrt{N}$ samples from a range of size N, the probability is about $1 - e^{-1/2} \approx 0.39$.

The birthday paradox explains that uniformly distributed birthdays across 365 days, denoted by $C_{q,365}$ the event that there is a collision among q values chosen independently from a set of 365 elements. From $\Pr[C_{q,365}] \geq \frac{1}{2}$ we obtain $q \equiv 23$, meaning the probability that two people out of 23 have a birthday on the same day is $\equiv 50\%$.

from calculus we get two facts;

1. $\forall\ x < 1 : 1 - x \leq e^{-x}$
2. $\forall\ 0 \leq x \leq 1 : e^{-x} \leq 1 - (1 - \frac{1}{e}) \cdot x \leq 1 - \frac{x}{2}$

Upper Bound Collision Probability: Let N be a positive integer and $(h_1, \ldots, h_q)$ are elements chosen independently and uniformly at random from a set of size N. Let $\Pr[C_{q,N}]$ denote the probability that there exists a collision pair $h_i = h_j$ for $i \neq j$. Then

$$\Pr[C_{q,N}] \leq \frac{q(q-1)}{2N}$$

Here N is the number of possible outputs of H i.e., $N = 2^l$.
Lower Bound Collision Probability: Let $q \leq \sqrt{2N}$, then

$$\frac{q(q-1)}{4N} \leq \Pr[C_{q,N}]$$

In summation, for $q \leq \sqrt{2N}$ we have $\frac{q(q-1)}{4N} \leq \Pr[C_{q,N}] \leq \frac{q(q-1)}{2N}$.

- A birthday attack finds a collision in time $O(2^{\frac{l}{2}})$ with constant probability for a random-looking $H : \{0,1\}^* \to \{0,1\}^l$. To reach probability about 1/2, one needs roughly $\sqrt{2N \ln 2}$ samples for $N = 2^l$ outputs.
- The complexity of $O(2^{\frac{l}{2}})$ is still exponential but much better than $O(2^l)$.
- To match security against brute-force search, we need to use H with $l = 2\lambda$-bit outputs.
- e.g., assume $H : \{0,1\}^* \to \{0,1\}^{128}$, then brute-force search runs in $O(2^{128})$, but a birthday attack runs in $O(2^{64})$.
- Note; birthday attacks do not work against PR2 or PR1 properties.

13.7 Random Oracle Model (ROM)

Since a hash function $H(m)$ is generally a public function, some security proofs idealize it as a random oracle rather than relying only on concrete properties such as CR, PR1, and PR2. This led to the Random Oracle Model (ROM).

- Formalised by Bellare and Rogaway in 1993
- Views hash functions as a random oracle
- makes security proofs of cryptographic schemes easier
- offers no guarantees if a standard hash function is used for implementation

$R(\cdot)$ on input m outputs a l-bit random string s. It is an infinite object and cannot be computed publicly by a PPT algorithm. To obtain $R(m)$, input m must be queried to the oracle $R(\cdot)$, whose outputs are unpredictable and uniformly distributed.

In the ROM, the idealized oracle $R(\cdot)$ is sampled as a truly random function; this is an assumption of the model, not a real-world guarantee about a concrete public hash function H. Inside that abstraction, an adversary learns the oracle value on m only by querying the oracle. If m is a new query, then choose a random $h \in \{0,1\}^l$ and save (m, h) in a table. If m was already queried, then look up h in the table and return it.

14 Hash Function Constructions

14.1 Length Extension Attacks on the Merkle-Damgard Transformation

Given $H(m)$ and $|m|$, a PPT adversary should not be able to produce $H(m\|m')$ for a message $m' \in \{0,1\}^*$ of its choice. The MD transformation is however susceptible to this form of attack, making MD4/5 and SHA0/1/2 all prone to length-extension attacks to varying degrees. SHA3 and BLAKE* are not vulnerable as they include defences against length-extension.

14.2 Davies-Meyer Compression Function

Original collision finding game:

$\mathsf{Game}^{\text{HashColl}}_{\mathcal{A},H}(\lambda)$:
  $s \xleftarrow{\$} \text{Gen}(1^\lambda)$
  $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, s)$
  if $m_0 \neq m_1$ and $H^s(m_0) = H^s(m_1)$ **then**
    return 1
  end
  return 0

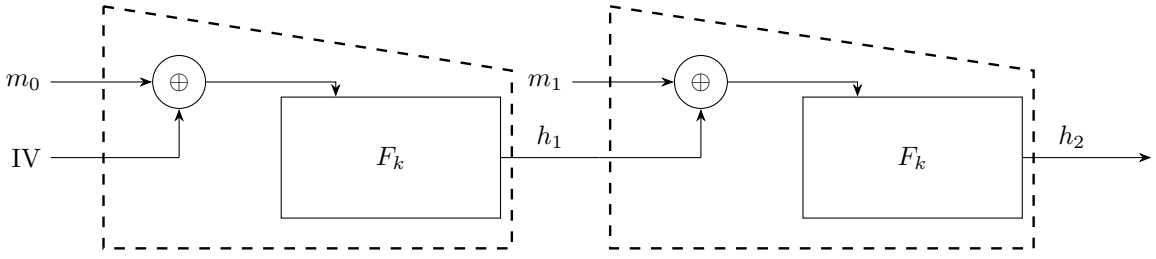
BC based collision finding game:

$\mathsf{Game}^{\text{HashBCColl}}_{\mathcal{A},H}(\lambda)$:
  $F \leftarrow \text{BlockCiphers}(\lambda, b)$
  $(m_0, m_1) \leftarrow \mathcal{A}^{F, F^{-1}}(\lambda, b)$
  if $m_0 \neq m_1$ and $H^s(m_0) = H^s(m_1)$ **then**
    return 1
  end
  return 0

DM construction: $f(h_i, m_i) := F_{m_i}(h_i) \oplus h_i$, In the original collision finding game, $\mathcal{A}$ gets the key s of H. In the BC game, finding the key is the block cipher F. $\mathcal{A}$ is given oracle access to $F_k(m)$ and $F_k^{-1}(c)$ for any k, m and c. A secure construction requires that;

$$\Pr[\mathsf{Game}^{\text{HashBCColl}}_{\mathcal{A},H}(\lambda) = 1] \leq \text{negl}(\lambda)$$

A weak compression function: Let $F : \{0,1\}^\lambda \times \{0,1\}^b \to \{0,1\}^b$ be a block cipher and k be the chosen public key.



This construction is susceptible to collision attacks.

Attack 1 we can create $m' \neq m$ such that $H^k(m') = H^k(m)$:

$$\begin{aligned} m &= \mathrm{IV} \| 0^b && \Rightarrow h_1 = F_k(0^b) && \Rightarrow h_2 = F_k(F_k(0^b)) \\ m' &= 0^b \| \, F_k(\mathrm{IV}) \oplus F_k(0^b) && \Rightarrow h_1 = F_k(\mathrm{IV}) && \Rightarrow h_2 = F_k(F_k(0^b)) \end{aligned}$$

Attack 2 For any k, IV, h_2 we can compute $m' \neq m$ such that $H^k(m') = H^k(m)$:

$$\begin{aligned} h_1 &= F_k^{-1}(h_2) \oplus m_1 && \Rightarrow z = F_k^{-1}(h_1) && \Rightarrow m_0 = z \oplus \mathrm{IV} \\ h_1' &= F_k^{-1}(h_2) \oplus m_1' && \Rightarrow z' = F_k^{-1}(h_1') && \Rightarrow m_0' = z' \oplus \mathrm{IV} \end{aligned}$$

14.3 Real Hash Functions

- **SHA-1**
  - Fingerprint length: 160 bits
  - Block length: 512 bits
  - Merkle-Damgard transformation
  - Davies-Meyer compression function built from $\neg, \wedge, \vee, \oplus, \ll, \lll$ operations
  - Rounds: 80
  - Designed by the NSA, published in 1995
  - PR1 and PR2: $O(2^{160})$
  - collisions: with birthday attacks $O(2^{80})$ **(unsafe)**
- **SHA-3 (Keccak)**
  - Output length: 224, 256, 384, 512 bits
  - Uses the sponge construction
  - SHA3-224/256/384/512 use rates $1152, 1088, 832, 576$ bits, respectively
  - Their capacities are $448, 512, 768, 1024$ bits, respectively
  - Uses Keccak-f[1600] with 24 rounds

Message Authentication Codes

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 27-31 Oct

Overview

- Padding oracles and chosen-ciphertext attacks (CCA)
- indistinguishability against chosen-ciphertext attacks (IND-CCA)
- Message integrity and authenticity
- Message authentication codes (MACs)
- Authenticated encryption (AE)
- Replay attacks and multi-user authentication

15 Padding Oracle

In CBC, if $|m|$ is not a multiple of the block size b, the last block must be padded. To recover the original message, padding must be easily identifiable and removable from the plaintext.

If incorrect padding is applied, the ciphertext will decrypt correctly, but the message will not be recovered. If failure to recover the message can be observed, this can leak information about the plaintext to an adversary that can manipulate the ciphertext.

15.1 Padding Oracle Attack on CBC

If we model the padding oracle as $\mathcal{O}_{\text{pad}}$. Given a ciphertext c, $\mathcal{O}_{\text{pad}}(c)$ returns 1 if the padding is correct and 0 otherwise. Let $m_{n,B}$ denote the last byte of m_n,$c_{n-1,B}$ denote the last byte of c_{n-1} and x_B the last byte of $F_k^{-1}(c_n)$. Then, $m_{n,B} = x_B \oplus c_{n-1,B}$.

The attacker can modify c_{n-1}, which changes the value of m_n. Specifically, the last byte $c_{n-1,B}$ to change $m_{n,B}$ which indicates how much padding was applied.

16 Indistinguishability under Chosen-Ciphertext Attack

Indistinguishability for the encrypted message m_b should hold even if $\mathcal{A}$ obtains plaintexts m_i for any ciphertext c_i of its choice at any time during the attack. Assuming Eve cannot just ask for the decryption of the target message, but can ask for the decryption of any other ciphertext. Can we abstract the padding oracle attack as a decryption oracle?

Definition

A private-key encryption scheme $\text{SE} = (\text{Gen}, \text{Enc}, \text{Dec})$ is IND-CCA if for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that:

$$\Pr[\mathsf{Game}_{\mathcal{A},\text{SE}}^{\text{IND-CCA}} = 1] \leq \frac{1}{2} + \text{negl}(\lambda)$$

Some Remarks:

- $\mathcal{A}$ has access to $\text{Enc}_k(\cdot)$ and $\text{Dec}_k(\cdot)$ oracles.
- $\text{Enc}_k(\cdot)$ oracle models chosen-plaintext attacks.
- $\text{Dec}_k(\cdot)$ oracle models chosen-ciphertext attacks.
- $\mathcal{Q}$ contains all ciphertexts that $\mathcal{A}$ queried to $\text{Dec}_k(\cdot)$ oracle.
- After the challenge is issued, the decryption oracle refuses the challenge ciphertext c, preventing $\mathcal{A}$ from trivially winning the game by decrypting it.

IND-CCA security game:

$\mathsf{Game}_{\mathcal{A},\text{SE}}^{\text{IND-CCA}}(\lambda)$:

$k \xleftarrow{\$} \text{Gen}(1^\lambda)$

$(m_0, m_1) \leftarrow \mathcal{A}^{\text{Enc}_k(\cdot),\text{Dec}_k(\cdot)}(1^\lambda)$

$b \xleftarrow{\$} \{0,1\}$

$c \leftarrow \text{Enc}_k(m_b)$

$b' \leftarrow \mathcal{A}^{\text{Enc}_k(\cdot),\text{Dec}_k(\cdot)}(c)$

return $b = b'$

16.1 Implication of IND-CCA to IND-CPA

Any IND-CCA secure encryption scheme is also IND-CPA secure.

Proof. Any adversary $\mathcal{A}$ that can win in the IND-CPA game can also win in the IND-CCA game. We show this with a simple reduction.

- Let $\mathcal{B}$ denote an IND-CCA adversary and let $\mathcal{A}$ denote an IND-CPA adversary.
- $\mathcal{B}$ answers to $\text{Enc}_k(\cdot)$ queries of $\mathcal{A}$ by forwarding m to its own oracle and by returning c to $\mathcal{A}$.
- $\mathcal{B}$ simply uses the same challenge message pair (m_0, m_1) that he receives from $\mathcal{A}$ and likewise forwards the challenge ciphertext c to $\mathcal{A}$.
- Clearly if $\mathcal{A}$ wins, then so does $\mathcal{B}$.
- Note: $\mathcal{B}$ does not use its $\text{Dec}_k(\cdot)$ oracle.

□

IND-CPA does not imply IND-CCA however.

Proof. We prove this by constructing a counterexample, a CCA attack breaks an IND-CPA secure scheme:

- $\text{Gen}(1^\lambda)$: Return $k \xleftarrow{\$} \{0,1\}^\lambda$
- $\text{Enc}_k(m)$: Pick $r \xleftarrow{\$} \{0,1\}^\lambda$ and return $c := (c_0, c_1) = (r, F_k(r) \oplus m)$
- $\text{Dec}_k(c)$: Return $m := F_k(c_0) \oplus c_1$.

The attack:

- $\mathcal{A}$ chooses $(m_0, m_1) = (0^\lambda, 1^\lambda)$ and asks for the challenge OTP with PRF ciphertext $c = (c_0, c_1)$
- $\mathcal{A}$ flips the first bit in c_1 via $(c_0, c_1 \oplus 10^{\lambda-1})$ and queries $\text{Dec}_k(c')$ to obtain m'.
- $\mathcal{A}$ outputs $b' = 0$ if $m' = 10^{\lambda-1}$ and $b' = 1$ if $m' = 01^{\lambda-1}$
- Note: that $\text{Dec}_k(\cdot)$ is required, and therefore must be CCA not CPA.

□

17 Message Integrity and Authentication

A quick reminder on second-preimage resistance; Consider Alice wanting to download a large file and a closer mirror server is offering faster download speeds. Alice obtains the fingerprint of the file, h, from the main server and checks if $H(\text{file}) = h$ to detect if the mirror sent the same file. If H is PR2, then this is enough to protect against a passive adversary.

However, if Eve actively interferes by a MITM attack replacing h with h', the adversary can pose as the mirror server and send m'. This example shows without authentication, we cannot protect integrity.

18 Message Authentication Codes (MAC)

A MAC consists of three algorithms:

- $k \leftarrow \text{Gen}(1^\lambda)$: Generates a (usually random) secret key k with $|k| \geq \lambda$.
- $t \leftarrow \text{Tag}(k, m)$: Generates an authentication tag t for a message m using key k.
- $\{0,1\} \leftarrow \text{Verify}(k, m, t)$: Returns 1 if tag t is valid with respect to m and k and 0 otherwise.

Correctness: For all $\lambda \in \mathbb{N}$, for all $k \leftarrow \text{Gen}(1^\lambda)$, and for all $m \in \{0,1\}^*$ it holds that:

$$\text{Verify}(k, m, \text{Tag}(k, m)) = 1$$

We see that Gen is probabilistic, Tag *can* be probabilistic, and Verify is always deterministic. If Tag is deterministic, $\text{Verify}(k, m, t)$ can be implemented by outputting 1 if and only if $t = \text{Tag}(k, m)$; this is called canonical verification (CV).

Unforgeability	Abbr.	Attacker goal
universal	UUF	Forge a tag for any message chosen by the challenger.
selective	SUF	Forge a tag for a message chosen by the attacker before the start of the game.
existential	EUF	Forge a tag for a message adaptively chosen by the attacker.

Table 1: Notions of security (Unforgeability)

Attack	Abbr.	Knowledge	Similar to
no-message attack	NMA	scheme definition	COA
known-message attack	KMA	learn tags for some messages	KPA
chosen-message attack	CMA	adaptively learn tags of any chosen messages	CPA
chosen-message attack w/ verification queries	CMVA	adaptively learn tags of any chosen messages and verify any tag-message pair	CCA

Table 2: Notions of security (Capabilities)

18.1 Notions of Security

Attacks against MACs (and digital signatures) have varying attacker capabilities and goals.
Attacker goal is always to forge a new tag for a message but the choice of message determines the strength of the unforgeability notion.
Attack Capabilities

By combining a goal and an attack, we obtain specific definitions for MAC security. Generally we can skip to prove strong existential unforgeability under chosen-message attacks (sEUF-CMA). By protecting against any message having a forged tag, an sEUF-CMA secure MAC can be as generic as possible (agnostic of the application or the content of the message). Capturing only that an adversary can modify and inject messages. Adversaries can also drop, re-order, reflect or replay messages, which require additional mechanisms to protect.

18.2 Existential Forgery with Chosen-Message Attack

Threat Model $\mathcal{A}$ can choose m_i and observe the corresponding tag t_i sent by Alice. $\mathcal{A}$'s goal is to forge a message pair (m, t) such that $\text{Verify}(k, m, t) = 1$ and m is a *new* message.

The message is considered new if $\mathcal{A}$ did not obtain the corresponding tag t from Alice. Otherwise $\mathcal{A}$ can just claim (m, t) as its own forgery. e.g., $\mathcal{A}$ is not allowed to re-use m for which it learned the corresponding tag t through its CMA.
Goal Adversary $\mathcal{A}$ should not be able to forge a tag t for a new message m of its choice.
Definition
A $\text{MAC} = (\text{Gen}, \text{Tag}, \text{Verify})$ provides EUF-CMA if for every PPT algorithm $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{MAC}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

Some Remarks

- $\mathcal{A}$ has access to $\text{Tag}_k(\cdot)$ oracle, modelling CMA.
- $\mathcal{Q}$ contains all messages m_i for which $\mathcal{A}$ obtained tags t_i through oracle queries.
- Though, EUF-CMA does not account for $\mathcal{A}$ forging a new tag $t_i' \neq t_i$ on a previously queried message $m_i \in \mathcal{Q}$.
- for sEUF-CMA, only difference is $\mathcal{Q}$ contains all (m_i, t_i), and $(m, t) \notin \mathcal{Q}$ for the security game.
- EUF-CMA MACs with canonical verification are sEUF-CMA.
- For strong MACs, access to the $\text{Verify}_k(\cdot)$ oracle does not help $\mathcal{A}$ asymptotically.
- Proofs involving MACs with CV can use EUF-CMA game.

MAC forgery game:

$\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{MAC}}(\lambda)$:
  $k \xleftarrow{\$} \text{Gen}(1^\lambda)$
  $(m, t) \leftarrow \mathcal{A}^{\text{Tag}_k(\cdot)}(1^\lambda)$
  if $\text{Verify}_k(m, t) = 1$ and $m \notin \mathcal{Q}$ **then**
    return 1
  end
  return 0

18.3 MACs and PRFs

Forging a MAC is comparable to computing a PRF without knowing the key.
Fixed-length MAC from PRF
Let $F_k : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a PRF and $\mathcal{M} \stackrel{\text{def}}{=} \{0,1\}^\lambda$.

- $\text{Gen}(1^\lambda)$: Return $k \xleftarrow{\$} \{0,1\}^\lambda$.
- $\text{Tag}(k, m)$: Return $t := F_k(m)$.
- $\text{Verify}(k, m, t)$: Return 1 if $t = F_k(m)$ and 0 otherwise.

Theorem If $F_k : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a PRF, then MAC with PRF F_k is EUF-CMA secure.
Proof Assume that this MAC is insecure and show that F_k is not a secure PRF.

18.4 Secure Variable Length MAC

A generic extension of Fixed-Length MAC; Let $(\text{Gen}', \text{Tag}', \text{Verify}')$ be a fixed length MAC.

- $\text{Gen} := \text{Gen}'$.
- $\text{Tag}_k(m)$: Let $l := |m|$, encode l and each block index i using fixed $\lambda/4$-bit encodings, and parse $m = m_1, \ldots, m_n$ with each $|m_i| = \frac{\lambda}{4}$. Pick $r \xleftarrow{\$} \{0,1\}^{\frac{\lambda}{4}}$. Return $t = (r, (t_1, \ldots, t_n))$ with $t_i = \text{Tag}'_k(r \| l \| i \| m_i)$.
- $\text{Verify}_k(m, t)$: Parse $t = (r, (t_1, \ldots, t_n))$, compute $l := |m|$, return 1 if $\text{Verify}'_k(r \| l \| i \| m_i, t_i) = 1$ holds for all $i \in \{1, \ldots, n\}$. Otherwise return 0.

We see that if r repeats for same-length m, m', then $\mathcal{A}$ can replace (m_i, t_i) with (m_i', t_i'). For large enough r, the probability that r repeats is negligible due to the birthday bound. If r does not repeat, $\mathcal{A}$ must forge (m_i, t_i) which is equivalent to an attack on MAC'.

This extension is inefficient however, $|m| = l \cdot \lambda$ bits implies $|t| = (4l + \frac{1}{4}) \cdot \lambda$ bits.

18.5 MACs and Hash Functions

By hashing message m using some secret key k, given any valid pair (m, t), CR would prevent $\mathcal{A}$ from finding message m' for which (m', t) is valid.

If we let $\text{Tag}_k(m)$ return $t := H(k \| m)$ and 0 otherwise. And then let $\text{Verify}_k(m, t)$ return 1 if $t = H(k \| m)$ and 0 otherwise.

We see that if CRHF uses the MD transformation, this construction is insecure as any pair (m, t) can produce a forgery with a length extension attack:

$$(m \| \text{ pad}(k \| m) \| m',\ H(k \| m \| \text{ pad}(k \| m) \| m'))$$

though this attack does not work if MD-finalization is used. This and other constructions (e.g., $H(m \| k)$ or $H(k \| m \| k)$) can be proved secure for MD hash functions under specific assumptions.

18.6 NMACs and HMACs

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a CRHF based on the MD transformation with a CR compression function $f : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$. The NMAC is specified as:

- $\text{Gen}(1^\lambda)$: Pick $k_1, k_2 \xleftarrow{\$} \{0,1\}^\lambda$ and return $k := (k_1, k_2)$.
- $\text{Tag}_k(m)$: Return $t := f(k_1 \| H^{k_2}(m))$ with IV $= k_2$ for H.
- $\text{Verify}_k(m,t)$: Return 1 if $t = \text{Tag}_k(m)$ and 0 otherwise.

The compression function f must act like a fixed-length MAC or a PRF. In general, the CR of f is not sufficient but modern compression functions (e.g., SHA-2) act like fixed-length MACs.

Theorem If f is CR and acts like a EUF-CMA-secure fixed-length MAC, then NMAC is EUF-CMA secure.

Proof Assume that NMAC is insecure. Show that f is not CR or that f does not act like a fixed-length MAC.

Assume $\mathcal{A}$ outputs a MAC forgery (m,t) and denote $\mathcal{Q} := \{m' | \mathcal{A} \text{ queried } \text{Tag}_k(m')\}$ Two cases are possible in this scenario:

- **Case 1** $\exists m' \in \mathcal{Q}$ such that $H^{k_2}(m) = H^{k_2}(m')$. This contradicts H being CRHF.
- **Case 2** $\forall m' \in \mathcal{Q}$ we have $H^{k_2}(m) \neq H^{k_2}(m')$. Then $(H^{k_2}(m), t)$ is a forgery for the fixed-length MAC given by f.

NMAC requires modification of the original IV used in H i.e., IV $= k_2$. This means that an implementation of H cannot be used in a black-box way to build NMAC.

Let H be a CRHF, ipad $=$ `0x36`$, \ldots,$ `0x36` and opad $=$ `0x5C`$, \ldots,$ `0x5C`. HMAC is specified as:

- $\text{Gen}(1^\lambda)$: Pick $k \xleftarrow{\$} \{0,1\}^\lambda$.
- $\text{Tag}_k(m)$: Return $t := H((k \oplus \text{opad}) \| H((k \oplus \text{ipad}) \| m))$.
- $\text{Verify}_k(m,t)$: Return 1 if $t = \text{Tag}_k(m)$ and 0 otherwise.

A special case of NMAC: $k_1 := f(IV \| (k \oplus \text{opad}))$, $k_2 := f(\text{IV} \| (k \oplus \text{ipad}))$, and $t = H^{k_1}(H^{k_2}(m))$.

19 Authenticated Encryption

A scheme that is IND-CCA protects against chosen-ciphertext attacks against confidentiality, but does not explicitly protect message integrity or authenticity. We need to adapt the definition of an encryption scheme to allow it to reject messages without integrity.

An authenticated encryption (AE) scheme defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ is a tuple of efficient algorithms (Gen, Enc, Dec) with

- Gen: $k \xleftarrow{\$} \mathcal{K}$
- Enc: $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$, $\text{Enc}_k(m) = c$
- Dec: $\mathcal{K} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$, where $\text{Dec}_k(c) = m$ if c is valid, otherwise $= \perp$.

The deterministic algorithm Dec outputs a message for valid ciphertexts or the error symbol $\perp$.

$$\text{Correctness: } \forall m \in \mathcal{M}, \forall k \in \mathcal{K} : \text{Dec}_k(\text{Enc}_k(m)) = m$$

19.1 Unforgeability of Authenticated Encryption

Definition A symmetric key encryption scheme $AE = (\text{Gen}, \text{Enc}, \text{Dec})$ has ciphertext integrity (INT-CTXT) if for all PPT $\mathcal{A}$ there exists a function negl such that:

$$\Pr[\mathsf{Game}^{\text{INT-CTXT}}_{\mathcal{A},\text{AE}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

Some Remarks

- Instead of forging a (m, t) pair, $\mathcal{A}$ has to forge a valid ciphertext c, similar to sEUF-CMA but for encryption.
- $\mathcal{A}$ has to forge a valid ciphertext c.
- $\mathcal{A}$ has access to the $\text{Enc}_k(\cdot)$ oracle.
- $\mathcal{Q}$ contains all ciphertexts that $\mathcal{A}$ queried to its $\text{Enc}_k(\cdot)$ oracle.
- We require that the output ciphertext $c \notin \mathcal{Q}$, preventing $\mathcal{A}$ from trivially winning by returning a ciphertext generated by the $\text{Enc}_k(\cdot)$ oracle.
- Ciphertext integrity implies plaintext integrity.

INT-CTXT security game:

$\mathsf{Game}^{\text{INT-CTXT}}_{\mathcal{A},\text{AE}}(\lambda)$:
  $k \xleftarrow{\$} \text{Gen}(1^\lambda)$
  $(c) \leftarrow \mathcal{A}^{\text{Enc}_k(\cdot)}$
  if $\text{Dec}_k(c) \neq \perp$ and $c \notin \mathcal{Q}$ **then**
    return 1
  end
  return 0

A symmetric-key encryption scheme $AE = (\text{Gen}, \text{Enc}, \text{Dec})$ is an authenticated encryption scheme (AE-secure) if it has indistinguishability under chosen-plaintext attack and ciphertext integrity.

- IND-CPA and INT-CTXT imply AE-secure.
- INT-CTXT can be obtained from sEUF-CMA-secure MAC.

If a symmetric-key encryption scheme $AE = (\text{Gen}, \text{Enc}, \text{Dec})$ is AE-secure, then it has indistinguishability under chosen-ciphertext attack.

AE-secure is a stronger notion than IND-CCA and can be built by carefully composing an IND-CPA secure symmetric encryption scheme and an sEUF-CMA secure MAC.

19.2 Encrypt-then-MAC

Let SE = (Gen, Enc, Dec) be an IND-CPA secure symmetric encryption scheme and let MAC = (Gen, Tag, Verify) be an sEUF-CMA secure MAC. Construct an AE scheme as follows:

- $\text{Gen}(1^\lambda)$: Return $k := (k_e, k_m)$ with $k_e := \text{SE.Gen}(1^\lambda)$ and $k_m := \text{MAC.Gen}(1^\lambda)$.
- $\text{Enc}_k(m)$: Return (c, t) with $c := \text{SE.Enc}_{k_e}(m)$ and $t := \text{MAC.Tag}_{k_m}(c)$.
- $\text{Dec}_k(c, t)$: Return $m := \text{SE.Dec}_{k_e}(c)$ if $\text{MAC.Verify}_{k_m}(c, t) = 1$ and an error $\perp$ otherwise.

19.3 Proof Reduction

Theorem Other AE variants exist (e.g., Encrypt-and-MAC and MAC-then-Encrypt) but they are not generally secure.

If SE is IND-CPA, and MAC is sEUF-CMA, then the Encrypt-then-MAC scheme is AE (and thus IND-CCA).

Proof Assume AE is not AE-secure, therefore, an adversary can break INT-CTXT or IND-CPA.

- INT-CTXT of AE reduces to sEUF-CMA of MAC.
- IND-CPA of AE reduces to IND-CPA of SE.

Case 1: Assume $\mathcal{A}$ wins by forging a ciphertext. Construct $\mathcal{B}$ against sEUF-CMA of MAC.

- Let $\mathcal{B}$ be an sEUF-CMA adversary against MAC and let $\mathcal{A}$ be an INT-CTXT adversary against AE.
- $\mathcal{B}$ has access to $\text{MAC.Tag}_{k_m}(\cdot)$ oracle and $\mathcal{B}$ has to simulate ciphertext integrity game to $\mathcal{A}$, including authenticated encryption oracle queries.
- $\mathcal{B}$ samples $k_e := \text{SE.Gen}(1^\lambda)$ as an encryption key that $\mathcal{B}$ uses when answering $\text{AE.Enc}_{k_e,k_m}(m)$ oracles queries. On an $\text{AE.Enc}_{k_e,k_m}(m)$ query from $\mathcal{A}$, $\mathcal{B}$ computes $c'_0 = \text{SE.Enc}_{k_e}(m)$ and queries its MAC oracle to get $t' = \text{MAC.Tag}_{k_m}(c'_0)$ and returns $c' = (c'_0, t')$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs its forgery c, which decrypts correctly if $\mathcal{A}$ can forge the tag.
- $\mathcal{B}$ parses c as (c_0, t) and outputs that as its forgery against MAC.
- Then;

$$\Pr[\mathcal{A} \text{ breaks AE-security with a MAC forgery}] \leq \Pr[\mathsf{Game}^{\text{sEUF-CMA}}_{\mathcal{B},\text{MAC}}(\lambda) = 1]$$

Case 2: Assume $\mathcal{A}$ wins the IND-CPA game by guessing the correct message. Construct $\mathcal{B}$ against the IND-CPA of SE.

- Let $\mathcal{B}$ be an IND-CPA adversary against SE and let $\mathcal{A}$ be an IND-CPA adversary against AE.
- $\mathcal{B}$ has access to $\text{SE.Enc}_{k_e}(\cdot)$ oracle and $\mathcal{B}$ has to simulate the IND-CPA game to $\mathcal{A}$, including ciphertext authentication.

- $\mathcal{B}$ samples $k_m := \text{MAC.Gen}(1^\lambda)$ as a MAC key that $\mathcal{B}$ uses when answering $\text{AE.Enc}_{k_e}(m)$ from its encryption oracle, computes $t' := \text{MAC.Tag}_{k_m}(c'_0)$.
- $\mathcal{A}$ outputs m_0, m_1, which $\mathcal{B}$ forwards to its IND-CPA challenger.
- $\mathcal{B}$ receives c, the encryption of m_0 or m_1, computes $t := \text{MAC.Tag}_{k_m}(c)$ and gives (c, t) to $\mathcal{A}$.
- When $\mathcal{A}$ outputs b', $\mathcal{B}$ forwards b' to its challenger.
- Then;

$$\Pr[\mathcal{A} \text{ breaks AE-security without a MAC forgery}] \leq \Pr[\mathsf{Game}^{\text{IND-CPA}}_{\mathcal{B},\text{SE}}(\lambda) = 1]$$

From both case reductions we get:

$$\Pr[\mathcal{A} \text{ breaks AE-security}] \leq \Pr[\mathsf{Game}^{\text{sEUF-CMA}}_{\mathcal{B},\text{MAC}}(\lambda) = 1] + \Pr[\mathsf{Game}^{\text{IND-CPA}}_{\mathcal{B},\text{SE}}(\lambda) = 1]$$

20 Replay Attacks and Multi-user Authentication

Replay attacks are a common threat to any secure communication protocol. The EUF-CMA security game for MACs does not capture replay attacks as it requires that $m \notin \mathcal{Q}$. MACs per-se do not provide protection against replay attacks. We must deal with replay attacks at the protocol level.

How can we tell if a message-tag pair (m, t) is fresh or a replay?

Solution 1: Session Identifiers

- During setup of a communication channel, parties agree on a unique session identifier *sid*.
- Parties compute tags as $t := \text{Tag}_k(sid \| i \| m_i)$ where i is a counter ensuring freshness and ordering in combination with *sid*, which also helps detect drop and re-order attacks.
- Protocols sometimes use different counter sequences to distinguish between parties' messages e.g., even numbers $i_A := 0, 2, 4, \ldots$ for Alice and odd numbers $i_B := 1, 3, 5, \ldots$ for Bob, which helps detect reflect attacks.

Solution 2: Timestamps

- Parties have synchronized clocks and compute tags as $t := \text{Tag}_k(s \| m_i)$ where s is a timestamp.
- Parties only accept messages if the tag is valid and the included timestamp is within a certain time frame.

20.1 MAC-based Authentication in Multi-User Settings

Systems where $n \geq 3$ users can communicate with each other and any user can send a message to any other user.

Challenge

- Assume all parties share the same MAC key k, then there is no way to distinguish who sent a given message-tag pair (m, t).
- Thus MACs with a single shared key do NOT provide sender authentication in a multi-user setting.

Solution

- Sender authentication in multi-user settings can be achieved if each pair of users i and j share an independent MAC key k_{ij}.
- The tag is then computed with the pairwise key k_{ij} bound to both sender i and receiver j.
- This solution is not scalable to large groups, though, as each party must manage $n - 1$ keys.

Computationally Hard Problems and Elliptic Curves

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 3-7 Nov

How can we build one-way functions from computationally hard problems?

21 Computationally Hard Problems

A problem is considered computationally hard if it cannot be solved efficiently which typically means "in polynomial time", This is related to the study of the $P \neq NP$ problem. It is unknown how to prove mathematically that a certain problem is hard (at least not for the useful ones). Instead we use reductions to relate the computational hardness of problems to each other. As a basis we start from certain problems that are assumed to be hard, e.g., for which no efficient algorithms are known despite many decades of research efforts.

21.1 Integer Factorization Problem (IFP)

As every positive integer $n \in \mathbb{N}$ can be uniquely expressed as $n = p_1^{e_1} \cdots p_m^{e_m}$ where $p_1, \ldots, p_m$ are unique primes and $e_1, \ldots, e_m \in \mathbb{N}$. Consider the special setting where $n = pq$ and p and q being λ-bit sized primes.

Given an integer $n = pq$ that's the product of two unknown primes p, q, the IFP is defined as the problem of finding the factorization of n. (i.e., can you find p and q just by knowing n)

There is currently no known (generic) algorithm that can factor n in polynomial time. Brute force methods currently include finding p by testing whether n/x is an integer for all primes $x \in [2, \ldots, n-1]$.

Complexity of factorisation is exponential: $O(2^{2\lambda} \cdot t)$ in λ the length of p and q where t is the time needed for division. Of course, we can improve this slightly to $O(2^{\lambda} \cdot t)$ by restricting $x \in [2, \ldots, \sqrt{n}]$.

21.2 The Rivest-Shamir-Adleman (RSA) Problem

Let $n = pq$ be a product of two large primes. The order of $\mathbb{Z}_n^*$ is $\phi(n) = (p-1)(q-1)$. If p and q are kept secret, computing $\phi(n)$ is hard.

Let $e \in \mathbb{N}$ with $\gcd(e, \phi(n)) = 1$. Then there exists a $d \in \mathbb{N}$ such that $ed \equiv 1 \pmod{\phi(n)}$, meaning d is the multiplicative inverse of $e \mod \phi(n)$.

When p, q are kept secret, $\phi(n)$ is also secret, then it is hard to compute d from e.

Trapdoor Functions

Function f is called a trapdoor function, if f is easy to evaluate but hard to invert without a special information.

Theorem

The map $f_e : x \mapsto x^e \mod n$ is a trapdoor function with trapdoor d under the assumption that it is hard to compute d from e and n for an unknown $\phi(n)$. The same holds for $f_d : x \mapsto x^d \mod n$ with respect to e.

Since $ed = 1 + k\phi(n)$ for some integer k, Euler's theorem (or CRT, for all residues modulo n) gives $x^{ed} \equiv x \pmod{n}$; hence $f_d(f_e(x)) = x$.

Integer Factorization and RSA

Definition

The RSA problem is *hard* relative to $\mathrm{Gen}_{\mathrm{RSA}}(\lambda)$ if for all PPT $\mathcal{A}$ there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\mathrm{RSA}}_{\mathcal{A},\mathrm{Gen}_{\mathrm{RSA}}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$$

RSA security game:

$\mathsf{Game}^{\mathrm{RSA}}_{\mathcal{A},\mathrm{Gen}_{\mathrm{RSA}}}$:
- $(n, e, d) \leftarrow \mathrm{Gen}_{\mathrm{RSA}(\lambda)}$
- $y \xleftarrow{\$} \mathbb{Z}_n^*$
- $x \leftarrow \mathcal{A}(n, e, y)$
- **return** $x^e \equiv y \mod n$

Given a PPT algorithm $\mathcal{B}$ that can factor n, we can construct a PPT algorithm $\mathcal{A}$ that can solve the RSA problem: $\mathcal{A}$ uses $\mathcal{B}$ to compute p and q then determines $\phi(n)$, d, and $x = y^d \mod n$. The RSA assumption depends on the modulus generator and the exponent e; invalid or poorly chosen exponents can make the problem trivial or insecure.

21.3 The Decisional Composite Residuosity Problem

Another Integer Factorization related hardness assumption is the DCR problem. DCR is hard relative to $\text{Gen}_{\text{DCR}(1^\lambda)}$ if for all PPT algorithms $\mathcal{D}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{D}(n, x^n \mod n^2) = 1] - \Pr[\mathcal{D}(n, y \mod n^2) = 1]| \leq \text{negl}(\lambda)$$

where $(n, \phi(n)) \leftarrow \text{Gen}_{\text{DCR}}(1^\lambda)$ and $x, y \xleftarrow{\$} \mathbb{Z}^*_{n^2}$.

In short, it is hard to decide whether for a given $z \xleftarrow{\$} \mathbb{Z}^*_{n^2}$ there exists an integer $x \xleftarrow{\$} \mathbb{Z}^*_{n^2}$ such that $z \equiv x^n \mod n^2$. The relative strength of these IF-based assumptions is DCR $\leq$ RSA $\leq$ IF.

21.4 The Discrete Logarithm (DL) Problem (DLP)

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order n and let $h \in \mathbb{G}$. The unique value $x \in \mathbb{Z}_n$ such that $g^x = h$ is the DL of h with respect to g and write $x = \log_g h$.

Definition

Given a cyclic group $\mathbb{G} = \langle g \rangle$ of order n and some element $h \in \mathbb{G}$, the DLP is defined as the problem of finding $x \in \mathbb{Z}_n$ such that $h = g^x$.

The DLP is easy to solve for small groups, e.g., Consider $p = 13$, $\mathbb{G} = \mathbb{Z}^*_{13}$ and generator $g = 6$. Then $\log_6 10 = 2 \in \mathbb{Z}_1 2$ since $6^2 \mod 13 = 10$. but hard when p is large, e.g., $|p| = 2048$ bits.

So how hard is it to actually compute $x = \log_g h \in \mathbb{Z}_n$?

Brute-Force by Trial Exponentiation

Find x by testing whether $g^x = h \forall x \in [0, \ldots, n-1]$, complexity being exponential in length of n; $O(2^{|n|})$.

21.5 The Discrete Logarithm Security Game

To prove security of schemes build from DLP, it is useful to formalize a security game, and adversarial advantage.

Let $\mathbb{G}$ be a cyclic group of order n with generator g. DLP is hard relative to $\text{Gen}_{\mathbb{G}}(1^\lambda)$ if $\forall$ PPT $\mathcal{A}$, there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\text{DL}}_{\mathcal{A},\text{Gen}_{\mathbb{G}}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

DLP security game:

$\mathsf{Game}^{\text{DL}}_{\mathcal{A},\text{Gen}_{\mathbb{G}}}(\lambda)$:
- $(\mathbb{G}, g, n) \leftarrow \text{Gen}_{\mathbb{G}}(1^\lambda)$
- $x \xleftarrow{\$} \mathbb{Z}_n$
- $y \leftarrow \mathcal{A}(\mathbb{G}, g, n, g^x)$
- **return** $x = y$

21.6 Computational Diffie-Hellman Problem

Let $\mathbb{G}$ be a cyclic group of order n with generator g. CDH is hard relative to $\text{Gen}_{\mathbb{G}}(1^\lambda)$ if $\forall$ PPT $\mathcal{A}$, there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\text{CDH}}_{\mathcal{A},\text{Gen}_{\mathbb{G}}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

CDH security game:

$\mathsf{Game}^{\text{CDH}}_{\mathcal{A},\text{Gen}_{\mathbb{G}}}(\lambda)$:
- $(\mathbb{G}, g, n) \leftarrow \text{Gen}_{\mathbb{G}}(1^\lambda)$
- $x, y \xleftarrow{\$} \mathbb{Z}_n$
- $h \leftarrow \mathcal{A}(\mathbb{G}, g, n, g^x, g^y)$
- **return** $h = g^{xy}$

In other words, CDH is defined as the problem of finding g^{xy} given elements $g, g^x, g^y \in \mathbb{G}$ for randomly chosen values $x, y \in \mathbb{Z}_n$. For certain groups, CDH problem is *assumed* to be hard such as cyclic subgroups $\mathbb{G} \subseteq \mathbb{Z}^*_p$ of prime order q, with p prime and q suitably chosen and sufficiently large. To date, the most efficient way to solve CDH is to solve the DL problem. No mathematical proof so far that the CDH or DL problems are indeed hard, expect in certain generic groups. A proof that either problem is hard will imply $P \neq NP$.

21.7 Decisional Diffie-Hellman Problem

Let $\mathbb{G}$ be a cyclic group of order n with generator g. DDH is hard relative to $\text{Gen}_{\mathbb{G}}(1^\lambda)$ if $\forall$ PPT $\mathcal{D}$, there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$|\Pr[\mathcal{D}(\mathbb{G}, g, n, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{D}(\mathbb{G}, g, n, g^x, g^y, g^z) = 1]| \leq \text{negl}(\lambda)$$

Meaning, DDH is defined as the problem of distinguishing (g^x, g^y, g^{xy}) from (g^x, g^y, g^z) for randomly chosen values $x, y, z \in \mathbb{Z}_n$.

21.8 DL-based Hardness Assumptions

- CDH assumption is considered to be a stronger assumption than the DL assumption because if one can compute x from g^x, then one can easily compute g^{xy} from g, g^x, g^y.
- DDH assumption is considered stronger than CDH, because if one can compute $g^x y$ from g, g^x, g^y one can easily distinguish (g^x, g^y, g^{xy}) from (g^x, g^y, g^z).
- In fact, there are certain groups where CDH is hard but DDH is easy, e.g., multiplicative groups Z_p^* where p is prime.

More formally:
CDH $\leq$ DL
Given a PPT $\mathcal{B}$ that can solve DL, we construct a PPT $\mathcal{A}$ that solves CDH: $\mathcal{A}$ uses $\mathcal{B}$ to compute x (without loss of generality y) and then computes $g^{xy} = (g^y)^x$ (or $(g^x)^y$).
DDH $\leq$ CDH
Given a PPT $\mathcal{B}$ that can solve CDH, we construct a PPT $\mathcal{D}$ that solves DDH: $\mathcal{D}$ uses $\mathcal{B}$ to compute g^{xy} and can then decide whether it was given g^{xy} or g^z.
Overall it holds that DDH $\leq$ CDH $\leq$ DL.

22 Elliptic Curves

DL, CDH and DDH assumptions are specified for finite cyclic groups $\mathbb{G}$. The multiplicative cyclic subgroup $\mathbb{G} \subseteq \mathbb{Z}_p^*$ of order q for sufficiently large values p and q are problematic as it does not provide sufficient security against algorithms such as GNFS allowing computation of DL in such groups in $\exp(\tilde{O}(\log p)^{1/3})$. GNFS was able to solve DLP modulo a 795-bit prime so $|p| \geq 2048$ bits in practice, and for high security applications, need even larger primes. Such large primes are computationally expensive and performance is impacted significantly.

An elliptic curve E is a group specified by a set of points $P = (x, y)$ over a base field $\mathbb{F}$ i.e., $x, y \in \mathbb{F}$ satisfying certain equations. Notation $\mathbb{G} = E(\mathbb{F})$. Common equations include:

- Weierstrass: $y^2 = x^3 + ax + b$
- Montgomery: $dy^2 = x^3 + ax^2 + x$
- Twisted Edwards: $ax^2 + y^2 = 1 + dx^2y^2$

Neutral element: the point at infinity $\mathcal{O}$
Main operations:

- Point addition: given $P, Q \in \mathbb{G}$ compute $P + Q \in \mathbb{G}$
- Scalar multiplication[2]: given $P \in \mathbb{G}$ and scalar $\alpha \in \hat{\mathbb{F}}$, compute $\alpha \cdot P \in \mathbb{G}$.

We call $\hat{\mathbb{F}}$ the scalar field, which is usually different from the base field $\mathbb{F} \neq \hat{\mathbb{F}}$

22.1 Point Addition

Consider Weierstrass curve in form

$$E : y^2 = x^3 + ax + b$$

with $a, b \in \mathbb{F}$ such that $-16(4a^3 + 27b^2) \neq 0$ and $\text{char}(\mathbb{F}) \neq 2, 3$.

- `add`(P, Q) computes $R = P + Q$ on E with $P = (x_p, y_p)$, $Q = (x_q, y_q)$ and $R(x_r, y_r)$.

This algorithm is not universal to other curve forms. This particular algorithm in practice leaks (timing) side-channel information.

[2]Assumed to be an OWF

22.2 Scalar Multiplication

The naive approach is to perform point addition α-times.
More efficiently, use *double-and-add* (requires $2\log 2(\alpha)$ additions)

- `mul` computes $R = \alpha \cdot P$ on E via *double-and-add*.

This algorithm also leaks (timing) side-channel information.

22.3 Point Properties

The negative of a point $P = (x, y) \in E(\mathbb{F})$ is specified as $-P = (x, -y)$. The *order* of a point $P \in E(\mathbb{F})$ is the smallest α such that $\alpha P = \mathcal{O}$, Notation: $\text{ord}(P) = \alpha$
Properties:

- $\text{ord}(\mathcal{O}) = 1$
- $\text{ord}(P) | \text{ ord}(E(\mathbb{F}))$
- If $|E(\mathbb{F})| = p$ then $\text{ord}(P) = p$ for all $P \in E(\mathbb{F})\backslash\{\mathcal{O}\}$
- If $|E(\mathbb{F})| = pq$ then $\text{ord}(P) \in \{p, q, pq\}$ for all $P \in E(\mathbb{F})\backslash\{\mathcal{O}\}$

22.4 Computational Hardness Assumptions

Integer Factorization and Discrete Log in multiplicative and elliptic curve groups provide interesting insights on the key sizes required to achieve certain security levels.

Security	**IF/MG-DL**	**ECC-DL**
128	3072	256
256	15360	512

Overall note - ECs offer generally better security guarantees and performance than traditional DL-based schemes.

Public-Key Encryption

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 10-14 Nov

What can we do so far? Alice and Bob can use some private-key encryption scheme $Enc_k(\cdot)$ to communicate securely if they already share a secret key k. However, what can they do if they do not already share a secret key?

23 PKE: The General Idea

Split the key into two parts, a public key pk, and a private key sk.

Then anyone can encrypt a message using the receiver's public key pk but such ciphertexts can only be decrypted by the corresponding private key sk.

23.1 PKE Schemes

- $\text{Gen}(1^\lambda)$: Generates a key pair $(\mathsf{sk}, \mathsf{pk})$
- $\text{Enc}(\mathsf{pk}, m)$: Encrypts a plaintext m using pk and outputs ciphertext c.
- $\text{Dec}(\mathsf{sk}, c)$: Decrypts a ciphertext c using sk and outputs a message m.

Any PKE scheme must satisfy correctness:

$$\forall \lambda \in \mathbb{N}, \forall (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(1^\lambda), \forall m \in \mathcal{M} : \text{Dec}(\mathsf{sk}, \text{Enc}(\mathsf{pk}, m)) = m$$

Note that Enc can be randomized but Dec is *always* deterministic. Gen must randomly pick key pairs from $\mathcal{K}$ according to some distribution. PKE is also often referred to as asymmetric encryption.

23.2 IND-CPA for PKE

Definition

A public-key encryption scheme $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ is indistinguishable against chosen-plaintext attacks (IND-CPA) if for every PPT adversary $\mathcal{A}$ there is a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\text{IND-CPA}}_{\mathcal{A},\text{PKE}}(\lambda) = 1] \leq \frac{1}{2} + \text{negl}(\lambda)$$

where the probability is taken over the randomness used by $\mathcal{A}$ and the randomness used in the security game.

IND-CPA game:

$\mathsf{Game}^{\text{IND-CPA}}_{\mathcal{A},\text{PKE}}(\lambda)$:

$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(\lambda)$

$(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})$

$b \xleftarrow{\$} \{0, 1\}$

$c \leftarrow \text{Enc}_{\mathsf{pk}}(m_b)$

$b' \leftarrow \mathcal{A}(c)$

return $b = b'$

$\mathcal{A}$ is given pk, so implicitly, he has encryption oracle access (this is not the case for private-key encryption). Note that IND-CPA is the weakest requirement for public-key encryption, and IND-CPA is not achievable if Enc is deterministic ($\mathcal{A}$ simply computes $c_0 = \text{Enc}_{\mathsf{pk}}(m_0)$ and compares if $c_0 = c$, then $b = 0$, else $b = 1$).

24 RSA Encryption

Recall - A function f is called a trapdoor function if f is easy to evaluate but hard to invert without a special information, the trapdoor. The function $f_e : x \mapsto x^e \mod n$ is a trapdoor function with trapdoor d under the assumption that it is hard to compute d from e and n for an unknown $\phi(n)$. The same holds for $f_d : x \mapsto x^d \mod n$ with respect to e.

Gen(λ):

- pick λ-bit primes p and q and compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
- Pick random $e \in [1, \phi(n) - 1]$ such that $\gcd(e, \phi(n)) = 1$ and compute its inverse d by satisfying $d \cdot e = 1 \mod \phi(n)$.
- outputs $\mathsf{sk} = (d, n), \mathsf{pk} = (e, n)$

$\mathbf{Enc}_{\mathsf{pk}}(m)$:

- We treat m as an element in $\mathbb{Z}_n^*$
- Output $c = f_e(m) = m^e \mod n$

$\mathbf{Dec}_{\mathsf{sk}}(c)$:

- Output $m = f_d(c) = c^d \mod n$

24.1 RSA Example and Chinese Remainder Theorem

Key Generation

- Let $p = 7, q = 13$.
- Then $n = 7 \cdot 13 = 91$, $\phi(91) = (6 \cdot 12) = 72$
- Let $e = 5$, then $\gcd(5, 72) = 1$ holds.
- Find d in $5d = 1 \mod 72 \Rightarrow d = 29$
- $\mathsf{sk} = (d, n) = (29, 91)$, $\mathsf{pk} = (e, n) = (5, 91)$

Encryption and Decryption

- To encrypt $m = 4$, calculate $c = 4^5 \mod 91 = 23$.
- To decrypt c, $m = 23^{29} \mod 91 = 4$.

CRT allows us to speed up *RSA decryption* by $\approx 75\%$. Compute $(1, x, y) = \text{egcd}(p, q), m_p = c^d \mod p = c^{d \mod p-1} \mod p$ and $m_q = c^{d \mod q-1} \mod q$ (note that $xp + yp = 1$). Returning $m = (m_p \cdot (yq \mod n) + m_q \cdot (xp \mod n)) \mod n$.

Example 1 consider the same parameters as before; $p = 7, q = 13, n = 91, e = 5, d = 29, c = 23$. Then

$$(1, 2, -1) = \text{egcd}(7, 13)$$

and

$$m_p = 23^{29 \mod 6} \mod 7 = 23^5 \mod 7 = 4,$$
$$m_q = 23^{29 \mod 12} \mod 13 = 23^5 \mod 13 = 4.$$

Finally,

$$m = (4 \cdot (-1 \cdot 13 \mod 91) + 4 \cdot (2 \cdot 7 \mod 91)) \mod 91 = (4 \cdot 78 + 4 \cdot 14) \mod 91 = 4$$

Example 2 Consider $p = 5, q = 11$, then $n = 5 \times 11 = 55$. Suppose we have

$$m_p = 3 \pmod 5, m_q = 7 \pmod{11}$$

CRT asks us to reconstruct $m \pmod{55}$ such that

$$m \equiv 3 \pmod 5, m \equiv 7 \pmod{11}$$

To find m, we start with the first congruence $m \equiv 3 \pmod 5$, so m has form $m = 3 + 5k$. Subtracting 3 from both sides we get $5k \equiv 4 \pmod{11}$.

Now, we need the inverse of $5 \pmod{11}$. Since:

$$5 \times 9 = 45 \equiv 1 \pmod{11}$$

The inverse of $5 \pmod{11} = 9$, Multiply both sides by 9:

$$k \equiv 4 \times 9 \pmod{11}$$
$$k \equiv 36 \pmod{11}$$
$$k \equiv 3 \pmod{11}$$

So taking $k = 3$, substitute into $m = 3 + 5k = 3 + 5(3) = 18$. We can check $18 \equiv 3 \pmod 5$ because $18 = 15 + 3$, and $18 \equiv 7 \pmod{11}$ because $18 = 11 + 7$.

24.2 RSA Attacks with $e = 3$

1. Short messages
   - If $m < \sqrt[3]{n}$, then $c = m^3 \mod n = m^3$ and $m = \sqrt[3]{c} \in \mathbb{Z}$.
2. Multiple Public Keys Using the Same e
   - If a sender encrypts the same message m to three receivers with $\mathsf{pk}_1 = (n_1, 3), \mathsf{pk}_2 = (n_2, 3), \mathsf{pk}_3 = (n_3, 3)$,and that n_1, n_2, n_3 are pairwise co-prime.
   - Then $m < \min\{n_1, n_2, n_3\}$ and $c_i = m^3 \mod n_i$ for $i \in \{1, 2, 3\}$.
   - We can let $n' = n_1 \cdot n_2 \cdot n_3 \cdot$, then compute $c' \leq n'$ such that $c' = c_1 \mod n_1, c' = c_2 \mod n_2$, and$c' = c_3 \mod n_3$ using CRT.
   - Since $c' = m^3 \mod n'$ and $m < \min\{n_1, n_2, n_3\}$, we have $m^3 < n'$
   - Therefore, m can be recovered by computing $\sqrt[3]{c'} \in \mathbb{Z}$.

24.3 RSA Messages

RSA encrypts messages $m \in \mathbb{Z}_n^*$. But messages are typically binary strings, how can we encode a message $m \in \{0,1\}^*$ as an element of $\mathbb{Z}_n^*$?

Let $l := |n|$ be the length in bits of n. Every binary string m of length $l - 1$ encodes an integer in $\mathbb{Z}_n$, but not necessarily an element of $\mathbb{Z}_n^*$. However, if the encoded message $m \in \mathbb{Z}_n \backslash \mathbb{Z}_n^*$, can we still decrypt it? In that case, $\gcd(m, n) \neq 1$. Since $n = pq$, there are three cases:

- $m = 0$: decryption works fine since $0^{ed} = 0$.
- $\gcd(m, n) = p$: then $\gcd(m, q) = 1$ and $m^{q-1} \equiv 1 \pmod{q}$. Decryption still works as $m^{ed} \mod n = m^{(p-1)(q-1)z+1} \mod n = m$
- $\gcd(m, n) = q$: similar to $\gcd(m, n) = p$.

24.4 Padded RSA

Textbook RSA is deterministic and therefore not IND-CPA. Adding ad hoc random padding is not, by itself, the standard secure construction; modern treatments use a precise padding/KEM construction such as OAEP or RSA-KEM. Let $\forall \lambda : l(\lambda) \leq 2\lambda - 2$, and $\mathcal{M} = \{0,1\}^{l(\lambda)}$, then:

- $\mathrm{Gen}(1^\lambda)$: return (n, e, d) as in $\mathrm{Gen}_{\mathrm{RSA}}$. note that $|n| = 2\lambda$
- $\mathrm{Enc}(n, e, m)$: pick $r \xleftarrow{\$} \{0,1\}^{|n|-l(\lambda)-1}$ and return $c = (r \| m)^e \mod n$.
- $\mathrm{Dec}(n, d, c)$: compute $m' = c^d \mod n$ and return $l(\lambda)$ lower bits of m'.

Security claims for RSA encryption require the exact construction and assumptions; the simple scheme above should be treated only as intuition for why deterministic textbook RSA must be randomized.

25 ElGamal Encryption

Motivation: For well-chosen cyclic groups $\mathbb{G}$, the mapping $x \mapsto g^x$ with $g \in \mathbb{G}$ is considered one-way, i.e., it is hard to determine x given g^x. We can use the DLP to build a cryptographic scheme.

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q. The ElGamal encryption scheme is as follows:

- $\mathrm{Gen}(1^\lambda) : x \xleftarrow{\$} \mathbb{Z}_q$, compute $y = g^x$ and return $(\mathsf{sk}, \mathsf{pk}) = (x, y)$
- $\mathrm{Enc}(\mathsf{pk}, m)$: map message m to an element in $\mathbb{G}$ and pick $r \xleftarrow{\$} \mathbb{Z}_q$, return $c = (c_1, c_2) = (g^r, y^r \cdot m)$
- $\mathrm{Dec}(\mathsf{sk}, c)$: return $m = c_2 \cdot c_1^{-x}$. (since $c_2 \cdot c_1^{-x} = (y^r \cdot m) \cdot (g^r)^{-x} = ((g^x)^r \cdot m) \cdot (g^r)^{-x} = m$)

E.g, Let $q = 11, p = 2 \cdot q + 1 = 23$ (where p and q are prime). Let $\mathbb{G}$ be the cyclic subgroup of $\mathbb{Z}_p^*$ of prime order q. The element $g = 3$ is a generator of $\mathbb{G}$ since $3^{11} \mod 23 = 1$. Then:

- $\mathrm{Gen}(\cdot)$: Let $\mathsf{sk} = x = 8 \in \mathbb{Z}_q$ and
$$\mathsf{pk} = y = g^x \mod p = 3^8 \mod 23 = 6 \in \mathbb{G}$$
- $\mathrm{Enc}(\mathsf{pk}, m)$: Let $m = 4 \in \mathbb{G}$ and $r = 7 \in \mathbb{Z}_q$. Then
$$c_1 = g^r \mod p = 3^7 \mod 23 = 2$$
and
$$c_2 = y^r \cdot m \mod p = 6^7 \cdot 4 \mod 23 = 12$$

- $\mathrm{Dec}(\mathsf{sk}, c)$: Compute

$$c_1^{-x} = 2^{-8} \mod 23 = (2^{-1})^8 \mod 23 = 8 \mod 23$$

Then

$$m = c_2 \cdot c_1^{-x} = 12 \cdot 8 \mod 23 = 4$$

Under the DDH assumption, ElGamal is IND-CPA. **Proof**: Assume ElGamal is not IND-CPA, solve the DDH problem.

Let $\mathcal{A}$ denote an adversary against ElGamal, let $\mathcal{D}$ denote an adversary against the DDH problem. As a challenge $\mathcal{D}$ gets either $g^z = g^{x_1 x_2}$ or $g^z = g^y$, then has to distinguish whether $c_2 = m_b \cdot h^{x_2}$ or $c_2 = m_b \cdot g^y$ (the latter is uniformly distributed in $\mathbb{G}$).

- by construction

$$\Pr[\mathcal{D}(\mathbb{G}, g, q, g^{x_1}, g^{x_2}, g^{x_1 x_2}) = 1] = \Pr[\mathsf{Game}^{\text{ind-cpa}}_{\mathcal{A},\text{ElGamal}} = 1]$$

- since c_2 is uniformly distributed in $\mathbb{G}$

$$\Pr[\mathcal{D}(\mathbb{G}, g, q, g^{x_1}, g^{x_2}, g^{y}) = 1] = \Pr[b' = b] = 1/2$$

-

$$\begin{aligned}\mathsf{Adv}^{\text{ddh}}_{\mathcal{D},\text{Gen}_\mathbb{G}}(1^\lambda) &= |\Pr[\mathcal{D}(\ldots, g^{x_1 x_2}) = 1] - \Pr[\mathcal{D}(\ldots, g^{y}) = 1]| \\ &= |\Pr[\mathsf{Game}^{\text{ind-cpa}}_{\mathcal{A},\text{ElGamal}} = 1] - 1/2| \\ &= \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\text{ElGamal}}(1^\lambda)\end{aligned}$$

- $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\text{ElGamal}}(1^\lambda) = \mathsf{Adv}^{\text{ddh}}_{\mathcal{D},\text{Gen}_\mathbb{G}}(1^\lambda) \leq \text{negl}(\lambda)$

25.1 ElGamal Messages

ElGamal encrypts messages m that are elements of the group $\mathbb{G}$.

- Let $p = 2q + 1$ where p and q are prime numbers with $l := |q|$. Let $g := h^2 \mod p$ for $h \xleftarrow{\$} \mathbb{Z}_p^* \backslash \{1, p-1\}$, then g generates a subgroup of order q in $\mathbb{Z}_p^*$.
- Let $x \in \{0,1\}^{l-1}$ be a binary string that we want to encode as an element of $\mathbb{G}$. Consider x as an integer in $[0, q-1]$ and compute $y = x + 1 \in [1, q]$. Compute $z = y^2 \mod p$ which ensures $z \in \mathbb{G}$.
- We can decode binary strings from $\mathbb{G}$ only after fixing a canonical square-root convention, since both y and $-y \bmod p$ square to the same value; after choosing the canonical root, output $x = y - 1$.

25.2 Public Parameters in ElGamal

$(\mathbb{G}, g, q) \leftarrow \text{Gen}_\mathbb{G}(1^\lambda)$ are public group parameters.

The DDH problem is hard relative to some $\text{Gen}_\mathbb{G}(1^\lambda)$. DDH is believed to remain hard even if $\text{Gen}_\mathbb{G}(1^\lambda)$ is run by the adversary as long as $(\mathbb{G}, g, q)$ satisfies the desired properties.

This means that $(\mathbb{G}, g, q)$ can be generated once and then used by multiple users. Each user i would have their own private-public key pair $(\mathsf{sk}_i, \mathsf{pk}_i) = (x_i, g^{x_i}) \in \mathbb{Z}_q \times \mathbb{G}$.

25.3 Hash-Based ElGamal

- $\text{Gen}(1^\lambda) : x \xleftarrow{\$} \mathbb{Z}_q$, compute $y = g^x$ and return $(\mathsf{sk}, \mathsf{pk}) = (x, y)$
- $\text{Enc}(\mathsf{pk}, m)$: pick $r \xleftarrow{\$} \mathbb{Z}_q$, and return $c = (c_1, c_2) = (g^r, m \oplus H(y^r))$
- $\text{Dec}(\mathsf{sk}, c)$: return $m = c_2 \oplus H(c_1^x)$. (since $c_2 \oplus H(c_1^x) = m \oplus H(y^r) \oplus H((g^r)^x) = m \oplus H(g^{xr}) \oplus H(g^{rx}) = m$)

Hash-based ElGamal is IND-CPA in ROM, assuming CDH is hard.

26 Homomorphic Public-Key Encryption

Let PKE = (Gen, Enc, Dec) be a public-key encryption scheme over some message and ciphertext spaces $\mathcal{M}$ and $\mathcal{C}$, respectively. Let $\cdot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ and $* : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$. We call PKE homomorphic if for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \text{Gen}(1^\lambda)$ with $\lambda \in \mathbb{N}$ and $m_1, m_2 \in \mathcal{M}$:

$$\text{Dec}(\mathsf{sk}, \text{Enc}(\mathsf{pk}, m_1) * \text{Enc}(\mathsf{pk}, m_2)) = m_1 \cdot m_2$$

26.1 Multiplicative Homomorphic Schemes

RSA:

$$\begin{aligned}\text{Enc}(n, e, m_1) \otimes \text{Enc}(n, e, m_2) &= m_1^e \mod n \cdot m_2^e \mod n \\ &= (m_1 \cdot m_2)^e \mod n\end{aligned}$$

ElGamal:

$$\begin{aligned}\text{Enc}(\mathbb{G}, g, q, h, m_1) \otimes Enc(\mathbb{G}, g, q, h, m_2) &= (g^{r_1}, m_1 \cdot h^{r_1}) \cdot (g^{r_2}, m_2 \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, (m_1 \cdot m_2) \cdot h^{r_1+r_2})\end{aligned}$$

26.2 Additively Homomorphic Schemes

Paillier:

$$\begin{aligned}\text{Enc}(n, m_1) \oplus \text{Enc}(n, m_2) &= (1+n)^{m_1} \cdot r_1^n \mod n^2 \cdot (1+n)^{m_2} \cdot r_2^n \mod n^2 \\ &= (1+n)^{m_1+m_2} \cdot (r_1 \cdot r_2)^n \mod n^2\end{aligned}$$

27 Cramer-Shoup

The definition of IND-CPA is weak if ciphertext integrity requires protection. $\mathcal{A}$ could try to obtain messages for ciphertexts of its choice at any time.

Indistinguishability for m_b should hold even if $\mathcal{A}$ obtains plaintexts m_i for any ciphertext c_i of its choice at any time during the attack. In practice, $\mathcal{A}$ can send some ciphertext c_i to Bob who decrypts it to m_i and then sends it to Alice or even replies to $\mathcal{A}$ directly, over multiple rounds.

27.1 IND-CCA Security for Public-Key Encryption

A public-key encryption scheme PKE = (Gen, Enc, Dec) is IND-CCA secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\text{IND-CCA}}_{\mathcal{A},\text{PKE}}(\lambda) = 1] - \frac{1}{2} \leq \text{negl}(\lambda)$$

Note that in $\mathsf{Game}^{\text{IND-CCA}}_{\mathcal{A},\text{PKE}}(\lambda)$, $\mathcal{A}$ has access to $\text{Dec}_{\text{sk}}(\cdot)$ oracle which models CCA. $\mathcal{Q}$ is the set containing all decryption queries by $\mathcal{A}$, preventing $\mathcal{A}$ trivially winning the game by decrypting c.

IND-CPA game:
$\mathsf{Game}^{\text{IND-CCA}}_{\mathcal{A},\text{PKE}}(\lambda)$:
$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}^{\text{Dec}_{\text{sk}}(\cdot)}(1^\lambda, \mathsf{pk})$
$b \xleftarrow{\$} \{0, 1\}$
$c \leftarrow \text{Enc}_{\mathsf{pk}}(m_b)$
$b' \leftarrow \mathcal{A}^{\text{Dec}_{\text{sk}}(\cdot)}(c)$
if $c \notin \mathcal{Q} \wedge b = b'$ **then**
  return 1
else
  return 0

27.2 Cramer-Shoup Encryption

A variant of ElGamal, CS provides ciphertext integrity protection and is therefore IND-CCA.

Let $\mathbb{G}$ be a cyclic group of prime order q with generators g_1 and g_2 and let $H : 0, 1^* \to \mathbb{Z}_q$ be a hash function. Then:

- $\text{Gen}(1^\lambda)$: Pick $x_1, x_2, y_1, y_2, z \xleftarrow{\$} \mathbb{Z}_q$ and compute $u_1 = g_1^{x_1} g_2^{x_2}$, $u_2 = g_1^{y_1} g_2^{y_2}$ and $u_3 = g_1^z$. Output private sk $= (x_1, x_2, y_1, y_1, Z)$ and public $pk = (u_1, u_2, u_3)$.

- $\mathrm{Enc}(\mathrm{pk}, m)$: Treat message m as an element in $\mathbb{G}$. Pick $r \xleftarrow{\$} \mathbb{Z}_q$ and compute $c_1 = g_1^r$, $c_2 = g_2^r$, $e = u_3^r \cdot m$, $v = u_1^r \cdot u_2^{r \cdot H(c_1, c_2, e)}$. Output $c = (c_1, c_2, e, v)$
- $\mathrm{Dec}(\mathrm{sk}, c)$: If $c_1^{x_1} \cdot c_2^{x_2} \cdot (c_1^{y_1} \cdot c_2^{y_2})^{H(c_1, c_2, e)} = v$ then output $m = e \cdot c_1^{-z}$ else output $\perp$.

Digital Signatures

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 17-21 Nov

How can Bob distinguish a message sent from Alice and Eve? The answer is in using message authentication. But is Alice able to dispute that she sent a certain message? Not if messages have non-repudiation.

Digital signatures (DS) are a concept from public-key cryptography with the goal of providing both sender authentication and non-repudiation (and therefore public verifiability, and transferability).

28 Components of Digital Signatures

A DS is a tuple of efficient algorithms (Gen, Sign, Verify) that are specified as:

- $\text{Gen}(\lambda)$: Generates a key pair $(\mathsf{sk}, \mathsf{pk})$
- $\text{Sign}(\mathsf{sk}, m)$: Signs m using sk and outputs signature σ.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Outputs 1 if σ is valid on m, otherwise 0.

Any DS scheme must satisfy the following correctness:

$$\forall \lambda \in \mathbb{N}, \forall (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(\lambda), \forall m \in \mathcal{M} : \text{Verify}(\mathsf{pk}, m, \text{Sign}(\mathsf{sk}, m)) = 1$$

28.1 Unforgeability of Digital Signatures

Definition
A DS is existentially unforgeable under chosen-message attacks (EUF-CMA) if for all PPT adversaries there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{DS}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

DS forgery game:

$\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{DS}}(\lambda)$:

$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(\lambda)$
$(m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}_{\mathsf{sk}}(\cdot)}(\lambda, \mathsf{pk})$
if $\text{Verify}_{\mathsf{pk}}(m, \sigma) = 1$ *and* $m \notin \mathcal{Q}$ **then**
  return 1
end
  return 0

$\mathcal{A}$ is given pk and can verify signatures produced with the corresponding sk. With signing oracle access (modelling CMA), $\mathcal{A}$ is able to query any m to $\text{Sign}_{\mathsf{sk}}(\cdot)$ and get a corresponding σ. $\mathcal{Q}$ contains all messages queried to $\text{Sign}_{\mathsf{sk}}(\cdot)$ to prevent a trivial forgery.

Subtlety with non-repudiation
Assume Alice sends (m, σ) to Bob who verifies that σ is valid and then shows (m, σ) to Eve. Alice can still dispute her signature if she makes sk_A public and then pretend that someone else misused it to sign m on her behalf.
So Non-repudiation is modeled by EUF-CMA as long as sk is secret.

29 RSA Signatures

Computing the private exponent d from $\mathsf{pk} = (n, e)$ is assumed hard. Therefore, from $c^d = m^{ed} \mod n = m$, we can use d as a signing key.

- $\text{Gen}(\lambda)$: Generates a key pair $\text{sk} = (n, d)$ and $\text{pk} = (n, e)$.
- $\text{Sign}(\mathsf{sk}, m)$: Signs m using sk and outputs signature $\sigma = m^d \mod n$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Outputs 1 if $\sigma^e \mod n = m$, otherwise 0.

RSA signatures do not provide EUF-CMA security (no-message attack, e.g.,): In the DS forgery game, $\mathcal{A}$ can produce a valid forgery (m, σ) without having control over m:

1. $\mathcal{A}$ picks any $\sigma \in \mathbb{Z}^*_n$ and computes $m = \sigma^e \mod n$,
2. (m, σ) is a valid forgery since $\mathcal{A}$ did not query m to the signing oracle.

$\mathcal{A}$ can use a multiplicative homomorphic property of RSA to produce a valid (m, σ) for any m it chooses.

Chosen Message Attack on RSA Signatures

1. Choose $m \in \mathbb{Z}_n^*$ and $m_1 \xleftarrow{\$} \mathbb{Z}_n^*$
2. Compute $m_2 = m \cdot m_1^{-1} \mod n$
3. Query m_1 to $\text{Sign}_{\mathsf{sk}(\cdot)}$ oracle to get σ_1
4. Query m_2 to $\text{Sign}_{\mathsf{sk}(\cdot)}$ oracle to get σ_2
5. Compute $\sigma = (\sigma_1 \cdot \sigma_2)$
6. Output (m, σ) as a forgery.

We see that (m, σ) is a valid forgery since $\mathcal{A}$ did not query m to its signing oracle $\text{Sign}_s k(\cdot)$ and $(\sigma_1 \cdot \sigma_2) \mod n = (m_1 \cdot m_2)^d \mod n = m^d \mod n$.

29.1 Full Domain Hash RSA Signatures

To get EUF-CMA for RSA signatures, we must remove the homomorphic property of the RSA function.

Let $H : \{0,1\}^* \to \mathbb{Z}_n^*$ with $\mathcal{M} = \{0,1\}^*$. The full domain hash RSA signature scheme is as follows.

- $\text{Gen}(\lambda)$: Generates a key pair $\text{sk} = (n, d)$ and $\text{pk} = (n, e)$.
- $\text{Sign}(\mathsf{sk}, m)$: Signs m using sk and outputs signature $\sigma = H(m)^d \mod n$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Outputs 1 if $\sigma^e \mod n = H(m)$, otherwise 0.

If H is CR, then it is hard to find $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$ Also, if H is OW, the previously described attack would not work. FDH-RSA is therefore EUF-CMA in ROM under RSA assumptions.

30 Hash-and-Sign Paradigm

Generalizing the FDH-RSA into a security paradigm.

Given an EUF-CMA signature scheme $\text{DS}' = (\text{Gen}, \text{Sign}, \text{Verify})$ over $\mathcal{M}$ and a CRHF $H : \{0,1\}^* \to \mathcal{M}$, we can construct an EUF-CMA scheme $\text{DS} = (\text{Gen}, \text{Sign}, \text{Verify})$ over $\{0,1\}^*$ as follows:

- $\text{Gen}(\lambda)$: Generates a key pair $(\mathsf{sk}, \mathsf{pk}) = \text{DS}'.\text{Gen}(\lambda)$
- $\text{Sign}(\mathsf{sk}, m)$: Signs m using sk and outputs signature $\sigma = \text{DS}'.\text{Sign}(\mathsf{sk}, H(m))$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Outputs $\text{DS}'.\text{Verify}(\mathsf{pk}, H(m), \sigma)$, otherwise 0.

H serves two roles, namely to enable processing of arbitrary sized messages by reducing their size and to "scramble" messages. The description of H is implicitly part of the scheme's description.

30.1 Security of Hash-and-Sign

If $\text{DS}' = (\text{Gen}, \text{Sign}, \text{Verify})$ is EUF-CMA, and H is CRHF, $\text{DS} = (\text{Gen}, \text{Sign}, \text{Verify})$ is EUF-CMA secure.

Assume that DS is insecure, show that DS' is insecure or that H is not CRHF. Let $\mathcal{A}$ be a PPT forger for DS that outputs a forgery (m, σ) at the end of the game and let Coll be an event that "message m' was queried to $\text{Sign}_s k(\cdot)$ and $H(m') = H(m)$"". Then:

$$\begin{aligned}\Pr[\mathsf{Game}_{\mathcal{A},\text{DS}}^{\text{EUF-CMA}}(\lambda) = 1] &= \Pr[\mathsf{Game}_{\mathcal{A},\text{DS}}^{\text{EUF-CMA}}(\lambda) = 1 \wedge \text{Coll}] \\ &\quad + \Pr[\mathsf{Game}_{\mathcal{A},\text{DS}}^{\text{EUF-CMA}}(\lambda) = 1 \wedge \neg\text{Coll}] \\ &\leq \Pr[\text{Coll}] + \Pr[\mathsf{Game}_{\mathcal{A},\text{DS}}^{\text{EUF-CMA}}(\lambda) = 1 \wedge \neg\text{Coll}]\end{aligned}$$

Conditioned on $\neg$Coll, the pair $(H(m), \sigma)$ is a valid forgery for the base scheme DS', so this term is bounded by $\Pr[\mathsf{Game}_{\mathcal{B},\text{DS}'}^{\text{EUF-CMA}}(\lambda) = 1]$ for the induced forger $\mathcal{B}$.

Thus $\Pr[\mathsf{Game}_{\mathcal{A},\text{DS}}^{\text{EUF-CMA}}(\lambda) = 1] \leq \Pr[\text{Coll}] + \Pr[\mathsf{Game}_{\mathcal{B},\text{DS}'}^{\text{EUF-CMA}}(\lambda) = 1]$, which is negligible.

31 Discrete Logarithm-Based Signatures

31.1 ElGamal Signature Scheme

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q and let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a CRHF. The (modified) Elgamal signature scheme is:

- $\text{Gen}(\lambda)$: Generates a key pair $(\mathsf{sk}, \mathsf{pk}) = (x \xleftarrow{\$} \mathbb{Z}_q, y = g^x)$
- $\text{Sign}(\mathsf{sk}, m)$: Signs m using sk and outputs signature $\sigma = (r, s)$. Pick $k \xleftarrow{\$} \mathbb{Z}_q^*$, and compute $r = g^k$ and $s = (H(m) - xr)k^{-1} \mod q$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Outputs 1 if $g^{H(m)} = y^r r^s$, otherwise 0.

Remarks: Textbook Elgamal was not hashed, so was not EUF-CMA. Correctness holds as $y^r r^s = (g^x)^r \cdot (g^k)^s = g^{xr+ks} = g^{xr+k(H(m)-xr)k^{-1}=g^{H(m)}}$. Commonly $\mathbb{G} \subseteq \mathbb{Z}_p^*$ with $|\mathbb{G}| = q$ and $p = 2q + 1$ for primes p and q.

Modified Elgamal signature scheme is asymptotically EUF-CMA secure if H is modelled as a random oracle. No concrete security reduction exists due to the abstract asymptotic proof in the ROM.

Proof Idea

- Assume $\mathcal{A}$ uses the signing oracle to obtain signature (r, s) for a message m. This is also a valid signature for a distinct message m' if $H(m) = H(m')$ which is a collision for CRHF H.
- Assume $\mathcal{A}$, without querying the signing oracle, can produce a forged signature (r, s) for a fixed message m. Then we can use $\mathcal{A}$ as a subroutine to construct an algorithm $\mathcal{B}$ to solve the discrete log problem. When running as a subroutine, $\mathcal{A}$ can be *forked* just before it queries the random oracle to produce the forgery, with the random oracle returning different values for $H(m)$ for the two executions (forking lemma). So $H(m) = h$ and $g^h = y^r \cdot r^s$ for one execution and $H(m) = h'$ and $g^{h'} = y^r \cdot r^{s'}$ for the other. Using the two equations, $\mathcal{B}$ can solve the discrete log problem with non-negligible probability.
- In either case, a PPT adversary existing leads to a contradiction.

ElGamal Variants

ElGamal can be rearranged as $H(m) = xr + ks \mod q$ and generalized to $u = xv + kw \mod q$ with $u, v, w \in \{H(m), r, s\}$, giving the following variants:

u	v	w	Sign	Verify
$H(m)$	r	s	$H(m) = xr + ks$	$g^{H(m)} = y^r r^s$
$H(m)$	s	r	$H(m) = xs + kr$	$g^{H(m)} = y^s r^r$
r	$H(m)$	s	$r = xH(m) + ks$	$g^r = y^{H(m)} r^s$
r	s	$H(m)$	$r = xs + kH(m)$	$g^r = y^s r^{H(m)}$
s	$H(m)$	r	$s = xH(m) + kr$	$g^s = y^{H(m)} r^r$
s	r	$H(m)$	$s = xr + kH(m)$	$g^s = y^r r^{H(m)}$

31.2 Digital Signature Standard (DSS)

The DSS, originally specified by NIST in 1994, with the latest version published in 2023, standardises the Digital Signature Algorithms. An RSA-based signature scheme based on PKCS #1 v2.1, and the ECDSA.

Historically, DSA was specified with parameters such as $|p| = 1024$ bits, $|q| = 160$ bits and SHA-1, but these are legacy parameters rather than a modern security recommendation. Current deployments use stronger approved parameter sets and hash functions.

DSA

- $\text{Gen}(\lambda)$: Pick $x \xleftarrow{\$} \mathbb{Z}_q$, compute $y = g^x \mod p$, return $(\mathsf{sk}, \mathsf{pk}) = (x, y)$
- $\text{Sign}(\mathsf{sk}, m)$: Pick $k \xleftarrow{\$} \mathbb{Z}_q^*$, compute $r = (g^k \bmod p) \bmod q$, and $s = (H(m) + xr)k^{-1} \bmod q$; retry if $r = 0$ or $s = 0$, otherwise return $\sigma = (r, s)$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$:
  - Parse σ as (r, s).
  - Then, check that $0 < r < q$ and $0 < s < q$ (else return 0);
  - Compute $w = s^{-1} \bmod q$, $u_1 = H(m)w \bmod q$, and $u_2 = rw \bmod q$;
  - Compute $v = ((g^{u_1} y^{u_2} \bmod p) \bmod q)$ and return 1 iff $v = r$.

Correctness:

$$\forall(\mathsf{sk}, \mathsf{pk}) = \text{Gen}(\lambda), \forall m \in \{0,1\}^* : \Pr[\text{Verify}(\mathsf{pk}, m, \text{Sign}(\mathsf{sk}, m)) = 1] = 1$$

31.3 Schnorr Signature Scheme

Let $(\mathbb{G}, g, q)$ be a cyclic group with $\mathbb{G} \subseteq \mathbb{Z}_p^*$ and safe prime $p = 2q + 1$. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a CRHF. The Schnorr signature scheme is specified:

- $\text{Gen}(\lambda)$: Pick $x \xleftarrow{\$} \mathbb{Z}_q$, compute $y = g^x \mod p$, return $(\mathsf{sk}, \mathsf{pk}) = (x, y)$
- $\text{Sign}(\mathsf{sk}, m)$: Pick $k \xleftarrow{\$} \mathbb{Z}_q$, compute $r = g^k \mod p$, and $c = H(r\|m)$, and $s = (xc + k) \mod q$ and return $\sigma = (s, c)$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Parse σ as (s, c); if $c = H(y^{-c} \cdot g^s \| m)$ return 1, otherwise 0.

Correctness holds as $y^{-c} \cdot g^s = g^{-xc} \cdot g^{xc+k} = g^k = r$ and $c = H(r\|m)$.
(key, nonce) pair reuse: If messages $m \neq m'$ are signed with sk and the same randomness $r = g^k$, then sk can be recovered from σ and σ':

$$\frac{s - s'}{c - c'} \bmod q = \frac{x \cdot (c - c') + k - k}{c - c'} \bmod q = x$$

Security of Schnorr

The Schnorr signature scheme is EUF-CMA secure in the ROM assuming the DLP is hard.
Proof:

- Let $\mathcal{A}$ be an adversary against the Schnorr signature scheme and let $\mathcal{B}$ be an adversary against the DLP.
- $\mathcal{B}$ answers $\mathcal{A}$'s $H(z)$ queries by checking if z is in the map of queries $\mathcal{Q}$. If z is new, $\mathcal{B}$ picks $h \xleftarrow{\$} \mathbb{Z}_q$ and stores (z, h) in $\mathcal{Q}$. Otherwise $\mathcal{B}$ retrieves the h that corresponds to z from $\mathcal{Q}$ and returns h.
- $\mathcal{B}$ answers $\mathcal{A}$'s $\text{Sign}_{\mathsf{sk}}(m)$ queries by simulating the signature algorithm:
  - picking $c, s \xleftarrow{\$} \mathbb{Z}_q$, computing $r := y^{-c} g^s$, storing $(r\|m, c)$ in $\mathcal{Q}$ (simulating a query to $H(\cdot)$) and returns (c, s).
- Upon receiving $(m, (c, s))$ from $\mathcal{A}$:
  - if $\text{Verify}(y, m, (c, s)) = 1$, $\mathcal{B}$ searches for $r\|m = y^{-c} g^s \| m$ in $\mathcal{Q}$.
  - $\mathcal{B}$ then rewinds the execution of $\mathcal{A}$ to just before the query $H(r\|m)$ that $\mathcal{A}$ used to generate the valid forgery, then forks $\mathcal{A}$ and replies with a different $c' \neq c$.
- Upon receiving $(m, (c', s'))$ from $\mathcal{A}$:
  - If $\text{Verify}(y, m, (c', s')) = 1$, $\mathcal{B}$ has forged signatures on message m with the same randomness k. Thus, $y^{-c} \cdot g^s = g^k = y^{-c'} g^{s'}$, and $\mathcal{B}$ can solve $-cx + s \equiv -c'x + s' \pmod q$ for x as $x = \frac{s'-s}{c'-c} \bmod q$.

32 One-Time Digital Signatures

EUF-CMA secure signature schemes assume that a key pair is used multiple times. But what if $(\mathsf{sk}, \mathsf{pk})$ is needed to sign only a single message?

- OTDS is any digital signature scheme where the same sk can be used only once.
- One-time signature schemes *can* be more efficient than regular schemes.
- EUF-CMA for OTDS changes to address this property:$\text{Sign}_1(\mathsf{sk}, \cdot)$ can be queried only once.

Theorem Any EUF-CMA secure signature scheme is also an EUF-CMA secure OTDS.

DS forgery game:
$\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{OTDS}}(\lambda)$:
  $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(\lambda)$
  $(m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}_1(\mathsf{sk},\cdot)}(\lambda, \mathsf{pk})$
  if $\text{Verify}_{\mathsf{pk}}(m, \sigma) = 1$ *and* $m \notin \mathcal{Q}$ **then**
    return 1
  end
    return 0

32.1 Merkle Signatures

Merkle Trees

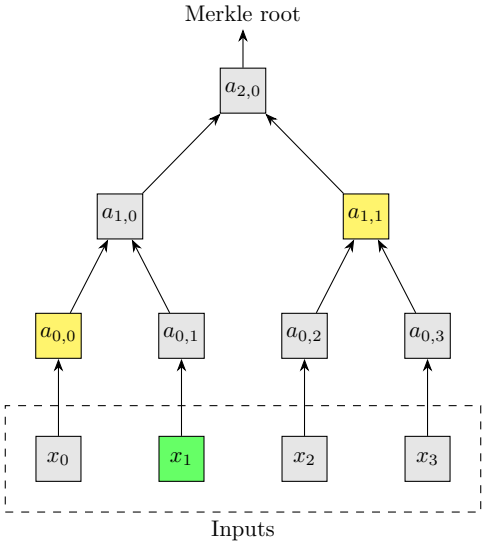
Let $\lambda \in \mathbb{N}$ and $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a CRHF and let $k \in \mathbb{N}$ and $n = 2^k$.

A (binary) MT over inputs $(x_0, \ldots, x_{n-1})$ with $x_i \in \{0,1\}^*$ is a binary hash tree of height $\log_2 n = k$ where nodes $a_{i,j}$ are computed as

$$a_{i,j} = \begin{cases} H(x_j) & \text{for } i = 0 \\ H(a_{i-1,2j} \,\|\, a_{i-1,2j+1}) & \text{for } 1 \leq i \leq k \end{cases}$$

where $0 \leq j \leq \frac{n}{2^i} - 1$. We call $a_{i,j}$ *leaves* for $i = 0$, *internal nodes* for $0 < i < k$, and *Merkle root* for $i = k$. A Merkle tree supports the following operations:

- Root$(x_0, \ldots, x_{n-1})$: Computes and returns the Merkle root r with respect to inputs $(x_0, \ldots, x_{n-1})$.
- Prove$((x_0, \ldots, x_{n-1}), j)$: Computes and returns the Merkle proof π for input x_j consisting of those k sibling nodes required to compute all the nodes between x_j and the Merkle root r.
- Verify(r, x_j, π): Returns 1 if π is valid with respect to x_j and r, and 0 otherwise.



e.g., the Merkle proof for x_1 in the tree above is $\pi = (a_{0,0}, a_{1,1})$ since $a_{2,0} = H(H(a_{0,0} \| H(x_1)) \| a_{1,1})$. A standard Merkle inclusion proof verifies that a certain input was present during construction of the tree; proving non-membership requires additional ordered-tree or non-membership machinery.

Merkle Signature Scheme

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a CRHF and OTDS $= (\text{Gen}, \text{Sign}, \text{Verify})$. Let $n = 2^k$ for some $k \in \mathbb{N}$ denote the expected number of messages. MSS is specified as:

- Gen(λ): Compute $(\mathsf{sk}_i, \mathsf{pk}_i) = \text{OTDS.Gen}(\lambda)$ and $r = \text{MT.Root}(\mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1})$. Return signing state containing $(\mathsf{sk}_0, \ldots, \mathsf{sk}_{n-1})$ together with the public keys/tree data needed to produce authentication paths, and public key $\mathsf{pk} = r$.
- Sign$((\mathsf{sk}_0, \ldots, \mathsf{sk}_{n-1}), m)$: Let m be the i-th message to be signed. Compute $s = \text{OTDS.Sign}(\mathsf{sk}_i, m)$ and $\pi_i = \text{MT.Prove}((\mathsf{pk}_0, \ldots, \mathsf{pk}_{n-1}), i)$ and return $\sigma = (s, \mathsf{pk}_i, \pi_i)$.
- Verify(pk, m, σ): Return 1 if $\text{MT.Verify}(\mathsf{pk}, \mathsf{pk}_i, \pi_i) = \text{OTDS.Verify}(\mathsf{pk}_i, m, s) = 1$ and 0 otherwise.

32.2 Chaining OTS

Assuming we do not know in advance how many messages we need to sign. We can use OTDS to sign multiple messages through a one-time signature chaining scheme.

- Gen(λ): Generate $(\mathsf{sk1}, \mathsf{pk1}) = \text{OTDS.Gen}(\lambda)$ and output $(\mathsf{sk}, \mathsf{pk}) = (\mathsf{sk1}, \mathsf{pk1})$.
- Sign(sk_i, m_i):
  - Generate $(\mathsf{sk}_{i+1}, \mathsf{pk}_{i+1}) = \text{OTDS.Gen}(\lambda)$,
  - compute $s_i = \text{OTDS.Sign}(\mathsf{sk}_i, \mathsf{pk}_{i+1} \| m_i)$,
  - store $(\mathsf{pk}_{i+1}, \mathsf{sk}_{i+1}, s_i, m_i)$ in state $\mathcal{S}$ and
  - return $\sigma_i = ((\mathsf{pk}_{i+1}, s_i, m_i), \ldots, (\mathsf{pk}_2, s_1, m_1))$.

- Verify$(\mathsf{pk}, m_i, \sigma_i)$: Return 1 if $\forall j = 1, \ldots, i$: OTDS.Verify$(\mathsf{pk}_j, \mathsf{pk}_{j+1} \| m_j, s_j) = 1$ and 0 otherwise.

33 Blind Signatures

If Bob wants a signature from Alice, but does not want Alice to know the content of m, the previous signing techniques will not be sufficient for this task.

33.1 Components of Blind Signature Schemes

Let U denote a user and S a signer. A BS scheme (Gen, Sign, Verify) consists:

- Gen(λ): Generates and returns a key pair $(\mathsf{sk}, \mathsf{pk})$ for S.
- Sign$(U(\mathsf{pk}, m); S(\mathsf{sk}, \mathsf{pk}))$: A protocol between U with input (pk, m) and S with input $(\mathsf{sk}, \mathsf{pk})$ returning a signature σ to U.
- Verify(pk, m, σ): Returns 1 if σ is a valid signature on m with respect to pk, otherwise 0.

BS has to fill correctness such that $\forall \lambda \in \mathbb{N}, \forall (\mathsf{sk}, \mathsf{pk}) \leftarrow \text{Gen}(\lambda), \forall m \in \mathcal{M}$, it holds that:

$$\text{Verify}(\mathsf{pk}, m, \text{Sign}(U(\mathsf{pk}, m); S(\mathsf{sk}, \mathsf{pk}))) = 1$$

33.2 Unforgeability of Blind Signatures

Definition A blind signature scheme BS = (Gen, Sign, Verify) is EUF-CMA if for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$\Pr[\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{BS}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

DS forgery game:
$\mathsf{Game}^{\text{EUF-CMA}}_{\mathcal{A},\text{BS}}(\lambda)$:
  $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(\lambda)$
  $(m_1, \sigma_1), \ldots, (m_{n+1}, \sigma_{n+1}) \leftarrow \mathcal{A}^{\text{Sign}_{\mathsf{sk}}(\cdot)}(\lambda, \mathsf{pk})$
  if $\forall i \in \{1, \ldots, n+1\}$: $\text{Verify}_{\mathsf{pk}}(m_i, \sigma_i) = 1$, $m_i \neq m_j$, *and* $|\mathcal{Q}| \leq n$ **then**
    return 1
  end
    return 0

We can see that the signing oracle $\text{Sign}_{\mathsf{sk}}(\cdot)$ starts a new execution of the BS.Sign$(\cdot)$ protocol on behalf of the signer. The number of BS.Sign$(\cdot)$ executions invoked by $\mathcal{A}$ is denoted as n. Also note that $\mathcal{A}$ must return $n + 1$ valid message/signature pairs with pairwise different messages $m_i \neq m_j$.

Blindness

Definition A blind signature scheme BS = (Gen, Sign, Verify) offers blindness for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that $\forall \lambda \in \mathbb{N}$:

$$|\Pr[\mathsf{Game}^{\text{blind}}_{\mathcal{A},\text{BS}}(\lambda) = 1] - \frac{1}{2}| \leq \text{negl}(\lambda)$$

DS forgery game:
$\mathsf{Game}^{\text{blind}}_{\mathcal{A},\text{BS}}(\lambda)$:
  $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \text{Gen}(1^\lambda)$
  $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk})$
  $b \xleftarrow{\$} \{0, 1\}$
  $\sigma_b \leftarrow \text{BS.Sign}(U(\mathsf{pk}, m_b)); S(\mathsf{sk}, \mathsf{pk})$
  $\sigma_{1-b} \leftarrow \text{BS.Sign}(U(\mathsf{pk}, m_{1-b})); S(\mathsf{sk}, \mathsf{pk})$
  $b' \leftarrow \mathcal{A}(\sigma_0, \sigma_1)$
  if $b' = b$ **then**
    return 1
  end
    return 0

$\mathcal{A}$ plays the role of the signer S and must distinguish the order in which m_0 and m_1 were signed. $\mathcal{A}$ wins ($b' = b$) if it can tell apart which message was signed in which of the two executions of the BS.Sign$(\cdot)$ protocol. Note that blindness protects the user U and not the signer S.

33.3 Chaum's BS Scheme

A variant of FDH-RSA (Section 29.1) enhancing it with the blindness property.

- $\text{Gen}(\lambda)$: Outputs $\mathsf{sk}, \mathsf{pk} = (n, d), (n, e)$
- $\text{Sign}(U(\mathsf{pk}, m); S(\mathsf{sk}, \mathsf{pk}))$:
  - U picks blinding factor $r \xleftarrow{\$} \mathbb{Z}_n^*$ and sends $s = H(m) \cdot r^e \mod n$ to the signer.
  - S replies with $t = s^d \mod n$.
  - U outputs the signature $\sigma = t \cdot r^{-1} \mod n$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: Output 1 if $H(m) = \sigma^e \mod n$, otherwise 0.

Note that $\sigma = H(m)^d \mod n$ is a regular FDH-RSA signature, that blindness holds since r hides $H(m)$ unconditionally. Unforgeability holds because FDH-RSA is EUF-CMA.

33.4 Blind Schnorr Signature Scheme

A variant of regular Schnorr (Section 31.3), enhancing them with the blindness property.

- $\text{Gen}(\lambda)$: Pick $x \xleftarrow{\$} \mathbb{Z}_q$, compute $y = g^x \mod p$, return $(\mathsf{sk}, \mathsf{pk}) = (x, y)$
- $\text{Sign}(U(\mathsf{pk}, m); S(\mathsf{sk}, \mathsf{pk}))$:
  - S picks $k \xleftarrow{\$} \mathbb{Z}_q$ and sends $r' = g^k \mod p$ to U.
  - U picks $a, b \in \mathbb{Z}_q$ computes $r = r' \cdot g^a \cdot y^b \mod p$ and $c = H(r \| m)$ and sends $c' = c + b \mod q$ to S.
  - S replies with $s' = x \cdot c' + k \mod q$
  - U then computes $s = s' + a \mod q$ and returns $\sigma = (s, c)$.
- $\text{Verify}(\mathsf{pk}, m, \sigma)$: If $c = H(y^{-c} \cdot g^s \| m)$ return 1, otherwise 0.

Correctness:

$$y^{-c} \cdot g^s = g^{-xc} \cdot g^{xc+xb+k+a} = g^{xb+k+a} = r' \cdot g^a \cdot y^b = r, \text{ and } c = H(r \| m)$$

34 Digital Certificates and PKI

Say Bob needs to verify Alice's signature on her message, he must know Alice's pk_A. Alice could send pk_A to Bob, but Eve intercepts it and replaces it with pk_E and can just impersonate Alice whenever communicating with Bob.

This problem can be solved by authenticated distribution of public keys.

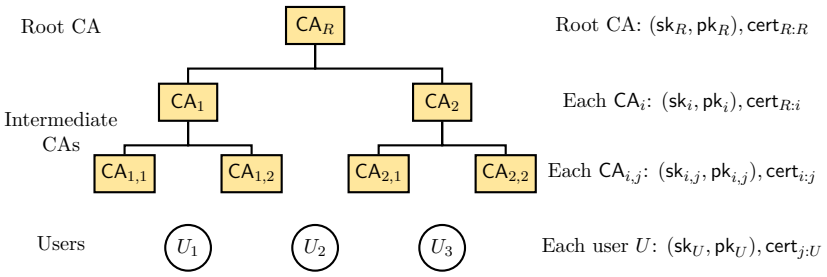
34.1 Digital Certificates

Assume there is a TTP that every user knows and trusts. Then the TTP can help to establish a connection between users in a secure and scalable manner.

Let $(\mathsf{sk}_T, \mathsf{pk}_T)$ be the private-public key pair of the TTP T and assume all users know and trust pk_T. We let $\text{DS} = (\text{Gen}, \text{Sign}, \text{Verify})$ be a EUF-CMA secure digital signature scheme. T can then issue a digital certificate to Alice for pk_A of the form:

$$\text{cert}_{T:A} \stackrel{\text{def}}{=} (\mathsf{pk}_A, \sigma_T = \text{Sign}(\mathsf{sk}_T, \text{"Alice's public key is } \mathsf{pk_A}\text{"}))$$

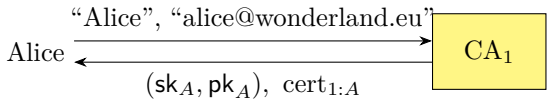
In practice there are multiple trusted third parties, the so-called certificate authorities (CA), which are arranged in a hierarchical manner forming the public key infrastructure (PKI).



A path $\text{cert}_{R:1}, \text{cert}_{1:1,1}, \text{cert}_{1,1:A}$ is called a certificate chain or chain of trust. For example, assume Alice has $((\mathsf{sk}_A, \mathsf{pk}_A), \text{cert}_{1,1:A})$ and sends $\text{cert}_{1,1:A}$ to Bob. To verify the received certificate $\text{cert}_{1,1:A}$, Bob must first check $\text{cert}_{R:1}$ and $\text{cert}_{1:1,1}$.

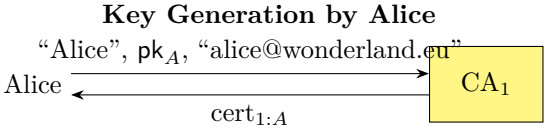
In practice $\text{cert}_{R:R}$ is universally trusted and, e.g., hard-coded into the system.

Key Generation by Trusted CA



This communication is over a secure channel and access to sk is typically protected with a password.

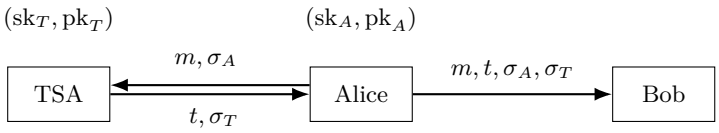
Key Generation by Alice



This communication is over an authenticated channel and Alice must prove knowledge of sk_A, e.g., by signing some message sent by CA_1.

Before issuing $\text{cert}_{1:A}$, CA1 performs several checks to verify Alice's identity (email verification, domain possession proof, personal identity card, etc.) and adds more information (CA identity, unique serial no. of cert, validity period, extensions).

34.2 Time Stamping



$\text{Verify}(\text{pk}_A, m, \sigma_A) = 1$

$\sigma_T := \text{Sign}(\text{sk}_T, m \,\|\, \sigma_A \,\|\, t)$

We of course want to link a signature σ_U by a user to some certain point in time t. Time stamps are DS's issued by a TTP Time-stamping Authority (TSA). Non-repudiation is guaranteed until sk_U has been made public.

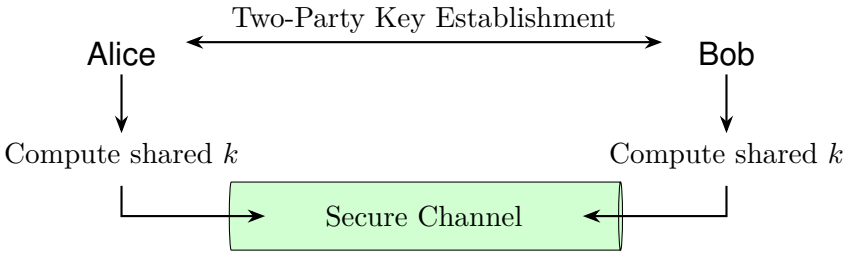
Key Establishment and Secret Sharing

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 17-21 Nov

Alice and Bob want to establish a shared secret key k over an insecure channel. They can use a secure key establishment protocol, which allows them to compute the same key k without directly transmitting it.

35 Secure Key Establishment



Properties of secure channels:

- Confidentiality: using symmetric encryption like AES, Alice and Bob can ensure that their communication is private and cannot be read by an eavesdropper.
- Authenticity: through MACs or digital signatures, Alice and Bob can verify the identity of the other party.

35.1 Users and Sessions

A session is a temporary, interactive information exchange between two or more users. Sessions are typically stateful, meaning, at least one of the communicating parties needs to hold and save state information to be able to communicate.

Any user U_i can start a new session with any other user U_j where $i, j \in \{1, \ldots, n\}$.

Sessions are identified through unique session identifier id that that can be computed through various means:

- $\mathsf{id} = r_i \| r_j$ for random numbers (nonces) r_i, r_j, chosen by U_i, U_j, respectively.
- $\mathsf{id} = t_{ij}$ where t_{ij} is the communication transcript of all incoming and outgoing messages between U_i and U_j.

35.2 Key Establishment Protocols

- **Key Transport**: The shared key is chosen by one party and needs to be sent to another party encrypted over an insecure channel.
  - Can either be through a trusted third party such as in kerberos
  - without a TTP; the TLS Key Transport.
- **Key Exchange**: The shared key is computed over an insecure channel as a function of two parties inputs (e.g., TLS Diffie-Hellman, IKE).

Properties of Key Establishment Protocols

- **Secrecy**: If user U_1 computes a shared key k with another user U_2, then k must remain secret to $\mathcal{A}$, i.e., $\mathcal{A}$ must not be able to decide whether k is real or random.
- **Authenticity**:
  - **One-sided**: Only one of U_1, U_2 must authenticate themselves during the KE protocol.
  - **Mutual**: Both U_1, U_2 must authenticate themselves during the KE protocol.

- **Key Agreement / Correctness**: If U_1 computes k_1 and U_2 computes k_2, then $k_1 = k_2$. **Key confirmation** is stronger: each party gains assurance that the peer actually possesses the same key.
- **Key Independence**: If the adversary $\mathcal{A}$ compromises the keys in some sessions, then this does not affect secrecy of the keys in other sessions.
- **Forward Secrecy**: If $\mathcal{A}$ corrupts some user U then shared keys that were computed by U in the past must remain secret. Note that $\mathcal{A}$ can impersonate corrupted users in future sessions.
- **Unknown Key Share (UKS) Resilience**: $\mathcal{A}$ must not be able to interfere in such a way that user U_1 believes that key k is shared with user U_2 although it is shared with some other user U_3. $\mathcal{A}$ does not need to actually learn k.
- **Key Compromise Impersonation (KCI) Resilience**: $\mathcal{A}$ must not be able to impersonate some honest user U_2 to a corrupted user U_1.

36 Key Transport Protocols

A Strawman: Let PKE = (Gen, Enc, Dec) be an IND-CCA secure public-key encryption scheme. Let $(\mathrm{sk}_B, \mathrm{pk}_B) = \mathrm{PKE.Gen}(\lambda)$ be an encryption key pair of Bob. Can a *passive* adversary $\mathcal{A}$ distinguish real k from random k^*? No, since $\mathcal{A}$ sees only c and PKE is IND-CCA secure. But, can an *active* adversary $\mathcal{A}$ attack the secrecy of the key computed by Bob? Yes, because $\mathcal{A}$ can pick $k' \xleftarrow{\$} \{0,1\}^\lambda$, then create $c' := \mathrm{Enc}_{\mathrm{pk}_B}(k')$ and send c' to Bob. This works as Alice does not authenticate herself to Bob.
Another Strawman: Let DS = (Gen, Sign, Verify) be a EUF-CMA digital signature scheme. Bob has an encryption key pair $(\mathrm{sk}_B, \mathrm{pk}_B) := \mathrm{PKE.Gen}(\lambda)$ and Alice has a signature key pair $(\mathrm{sk}_A, \mathrm{pk}_A) := \mathrm{DS.Gen}(\lambda)$. Now, can an *active* adversary $\mathcal{A}$ attack the secrecy of the key computed by Bob? Yes, as $\mathcal{A}$ can replay $c\|\sigma$ and break key independence. This works because Bob cannot check whether $c\|\sigma$ is a *fresh* message.

36.1 Replay Attack Prevention via Nonces

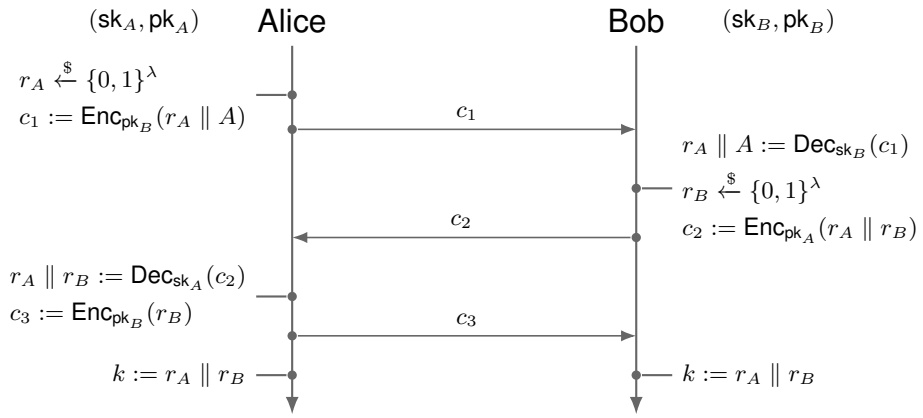
A nonce is a random number used once and chosen freshly for each new session. Can an *active* adversary $\mathcal{A}$ replay $r_A\|c\|\sigma$ in another session? Since verification of σ will fail unless r_B is the same in that other session, no. Consider length $\lambda = |r_B|$, the probability that Bob chooses the same nonce r_B in q sessions is (negligible)

$$\Pr[r_B \text{ repeats in } q \text{ sessions}] \approx \frac{q(q-1)}{2^{\lambda+1}}$$

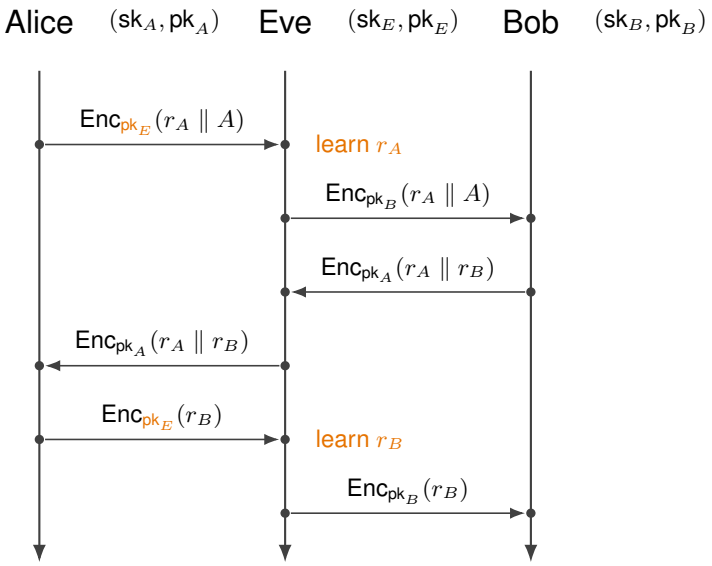
What if $\mathcal{A}$ corrupts Alice or Bob and learns their static private keys? If $\mathcal{A}$ learns sk_A, the secrecy of k is still protected. If $\mathcal{A}$ learns sk_B, then it can decrypt c and distinguish k. Assume that Alice and Bob have static signing key pairs $(\mathrm{sk}_A, \mathrm{pk}_A)$ and $(\mathrm{sk}_B, \mathrm{pk}_B)$ respectively, that they use for authentication. Forward secrecy can be achieved if Bob chooses a fresh one-time encryption key pair $(\mathrm{sk}'_B, \mathrm{pk}'_B)$ for every session he shares with Alice.

36.2 Needham-Schroeder PKE-Based Key Transport (NS-PKE-KT)

Assume Alice and Bob have PKE key pairs $(\mathrm{sk}_A, \mathrm{pk}_A)$ and $(\mathrm{sk}_B, \mathrm{pk}_B)$ respectively.



An attacker Eve can trick Alice to establish a session with her on behalf of Bob, Eve can relay Alice's message to Bob and convince him that he is communicating with Alice.



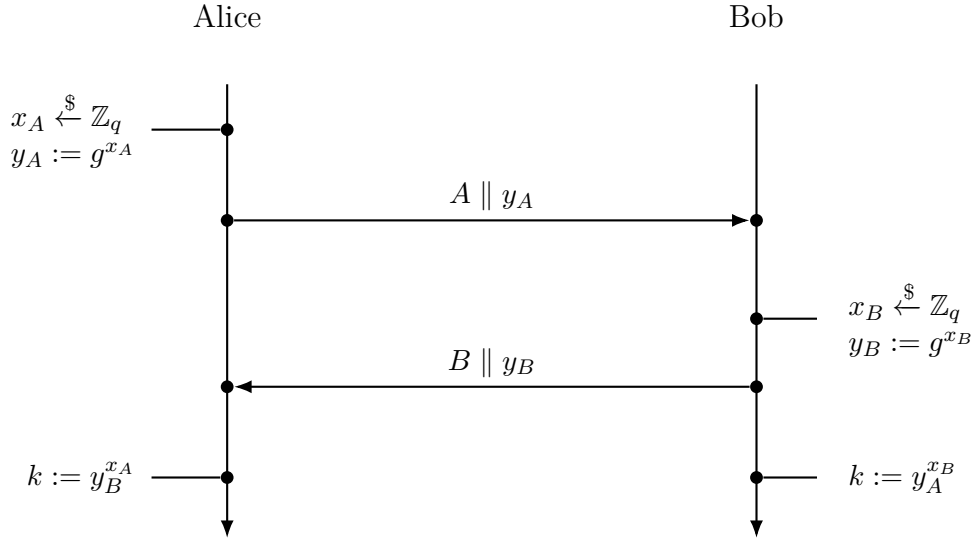
To prevent this attack change the second message from $\mathrm{Enc}_{\mathrm{pk}_A}(r_A \| r_B)$ to $\mathrm{Enc}_{\mathrm{pk}_A}(r_A \| r_B \| B)$. After Eve relayed this message, Alice will abort because she will find Bob's identifier B but expected to find Eve's identifier E. So Eve will not be able to learn r_B.

Key transport protocols use PKE schemes to transport the session key k. We can think of this as an application of two permutations p_B and p_B^{-1} chosen by Bob to k chosen by Alice. We can avoid using p_B^{-1} if we can think of two one-way functions f_A and f_B such that the shared secret value is a function of secrets x_A and x_B.

37 Key Exchange Protocols

37.1 Diffie-Hellman Key Exchange (DHKE)

Let $(\mathbb{G}, g, q)$ be a cyclic group of prime order q generated by g. To establish a shared key via DHKE, Alice and Bob proceed as follows.



The shared key $k = y_A^{x_B} = y_B^{x_A} = g^{x_A x_B}$ is an element of the group $\mathbb{G}$.

Example
Let $\mathbb{G} = \langle 3 \rangle$ be a subgroup of $\mathbb{Z}_{23}^*$ of prime order $q = 11$. To establish a shared DH key, Alice and Bob proceed:

1. Alice picks random $x_A \xleftarrow{\$} 7$ and computes $y_A = 3^7 \pmod{23} = 2$ and sends $A\|2$ to Bob.
2. Bob receives Alice's message and picks random $x_B \xleftarrow{\$} 4$, computes $y^B = 3^4 \pmod{23} = 12$ and sends $B\|12$ to Alice.
3. Finally, Alice computes $k = 12^7 \pmod{23} = 16$ and Bob computes $k = 2^4 \pmod{23} = 16$ as the shared key.

Note: The shared key $k = 3^{7\cdot 4 \pmod{11}} \pmod{23} = 3^6 \pmod{23} = 16$ is an element of group $\mathbb{G}$.

Session keys computed by DHKE are indistinguishable from random elements of $\mathbb{G}$ for any passive PPT adversary $\mathcal{A}$ under DDH assumptions.

Proof

Let $\mathcal{A}$ be a passive adversary against DHKE, and let $\mathcal{D}$ be an adversary against DDH. We show that if $\mathcal{A}$ can distinguish the real DHKE session key $k = g^{x_1 x_2}$ from a random element $k^* = g^y$, then $\mathcal{D}$ can solve DDH.

If $z = x_1 x_2 \pmod{q}$ then $k = g^{x_1 x_2}$ is the real DHKE session key and $\mathcal{A}$ outputs $b = 1$. If $z = y$ then $k = g^y$ is some random element from $\mathbb{G}$ and $\mathcal{A}$ outputs $b = 0$.

Therefore, if $\mathcal{A}$ can distinguish between the real k and a random k^*, then $\mathcal{D}$ can solve DDH.

Key Derivation Functions (KDFs)

In most situations, we cannot use group elements as symmetric keys directly. e.g., AES secret key is 128-bits. A KDF takes as input the secret group element next to some auxiliary information potentially and maps it to a bit string of the required size.

KDFs from Hash Functions

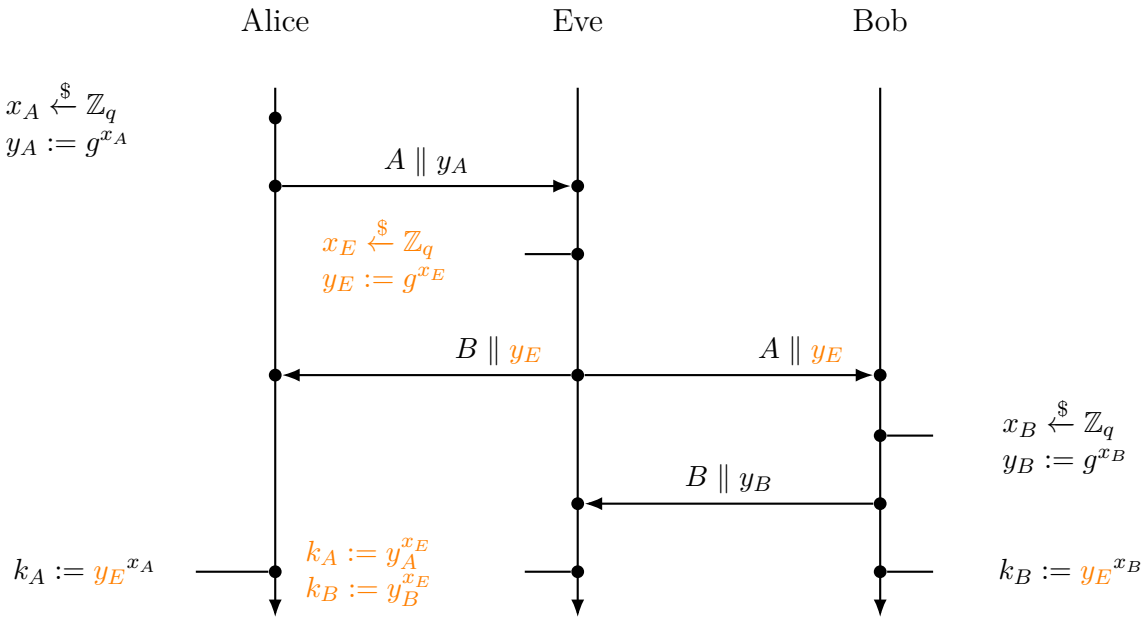
Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a CRHF. Then compute the DH session key as $k = H(A, y_A, B, y_B, g^{x_A x_B})$. If H behaves like a random oracle, then k is a uniformly distributed binary string.

KDFs from PRFs

Let $F_s : \{0,1\}^* \to \{0,1\}^\lambda$ be a PRF with key $s \in \{0,1\}^\lambda$. A proper extractor/KDF should first derive s from the DH group element and transcript; simply taking the most significant bits of $g^{x_A x_B}$ is not generally justified. Then compute the session key as $k := F_s(A, y_A, B, y_B)$.

37.2 Man-in-the-Middle Attack on DHKE

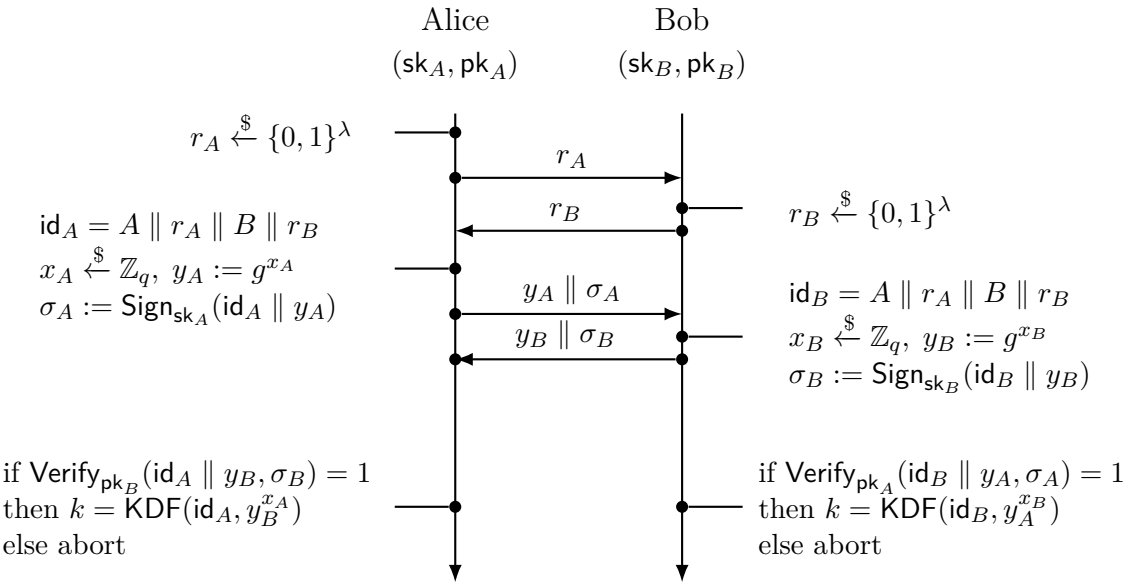
If $\mathcal{A}$ is an active adversary then DHKE becomes insecure due to a simple MitM attack.



In this case, Alice and Bob end up computing different group elements $k_A \neq k_B$ and $\mathcal{A}$ will know both k_A, k_B, allowing decryption and encryption of all messages between Alice and Bob.

37.3 Signed Diffie-Hellman Key Exchange

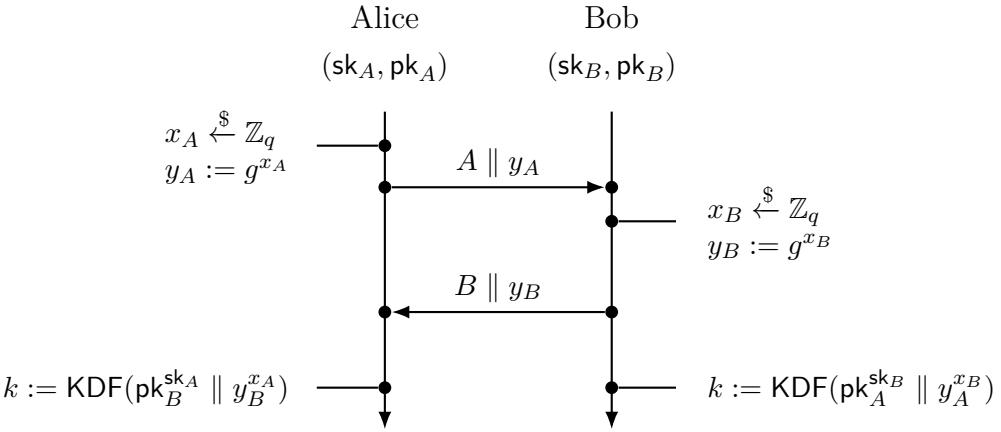
Assume Alice and Bob have signing key pairs $(\mathsf{sk}_A, \mathsf{pk}_A), (\mathsf{sk}_B, \mathsf{pk}_B)$, respectively.



If r_A or r_B is modified, then $id_A \neq id_B$ and verification of σ_A or σ_B fails. The above approach can be generalized into an authentication compiler to transform any KE protocol that is secure against a passive adversary into a KE protocol secure against an active one.

Unified Model Key Exchange (UMKE)

Authentication in KE protocols can be achieved using signatures. However, signature generation and verification can be costly.



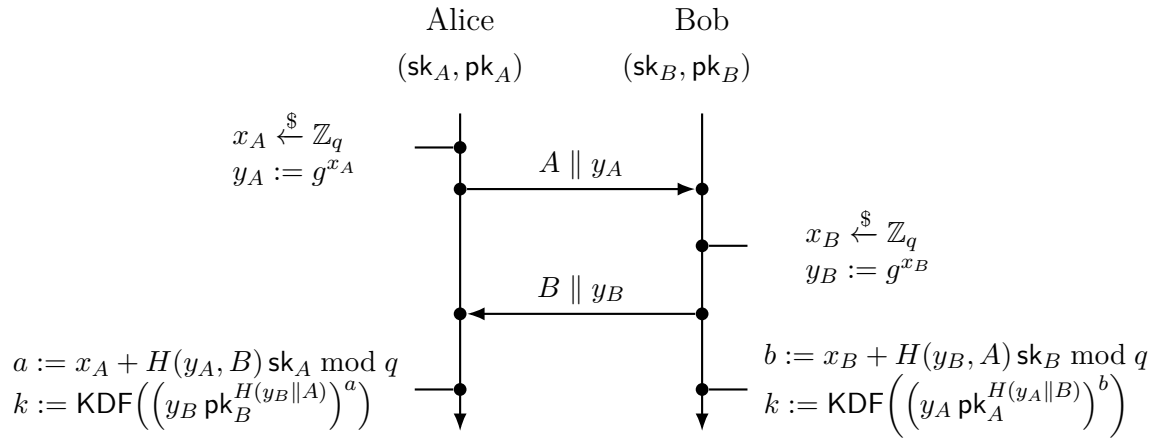
Assume Alice and Bob hold static DH key pairs $(\mathsf{sk}_A, \mathsf{pk}_A) = (s_A, g^{s_A})$ and $(\mathsf{sk}_B, \mathsf{pk}_B) = (s_B, g^{sB})$, respectively. The shared session key k is derived from an ephemeral value $g^{x_A x_B}$ and the static value $g^{s_A s_B}$ which only Alice and Bob can compute. This approach provides implicit authentication.

UMKE-based protocols are vulnerable to Key Compromise Impersonation attacks. Say that Eve learns sk_A of Alice, Eve can impersonate Bob to Alice without knowing Bob's sk_B:

1. Alice picks an ephemeral secret $x_A \xleftarrow{\$} \mathbb{Z}_q$, computes $y_A = g^{x_A}$ and sends y_A to "Bob" (aka. Eve).
2. Eve then picks an ephemeral secret $x_E \xleftarrow{\$} \mathbb{Z}_q$, computes $y_E = g^{x_E}$ and sends y_E to Alice.
3. Alice derives $k = \mathrm{KDF}(\mathsf{pk}_B^{\mathsf{sk}_A} \| y_E^{x_A})$
4. Eve derives $k = \mathrm{KDF}(\mathsf{pk}_B^{\mathsf{sk}_A} \| y_A^{x_E})$

HMQV Key Exchange

HMQV is resistant to KCI attacks. Its analysis requires H to behave like a random oracle or PRF-like hash function, not merely a collision-resistant hash function. HMQV works as follows:



The input to the KDF is of the form $g^{(x_A+H(y_A\|B)\mathsf{sk}_A)(x_B+H(y_B\|A)\mathsf{sk}_B)}$. It includes elements $g^{x_Ax_B}$ and $g^{\mathsf{sk}_A\mathsf{sk}_B}$ similar to UMKE but in addition, involves $g^{x_A\mathsf{sk}_B}, g^{x_B\mathsf{sk}_A}$.

Proof
A reminder of Key Exchange values:

- sk_i: long-term static secret key
- pk_i: long-term public key where $\mathsf{pk}_i = g^{\mathsf{sk}_i}$
- x_i: fresh ephemeral secret exponents chosen for a single session.
- y_i: corresponding ephemeral public values where $y_i = g^{x_i}$
- a : Alice's effective secret exponent $a = x_A + H(y_A, B)\mathsf{sk}_A$
- K_A: group element which Alice computes as the shared secret before applying the KDF.

Let Eve compromise Alice's sk_A. Alice sends $y_A = g^{x_A}$, and Eve responds $y_E = g^{x_E}$.

In HMQV, Alice does not compute the key from y_E alone. she computes

$$a = x_A + H(y_A, B)\mathsf{sk}_A$$

and derives the shared group element as

$$K_A = (y_E \cdot \mathsf{pk}_B^{H(y_E,A)})^a$$

Since $\mathsf{pk}_B = g_B^{sk}$, this becomes

$$K_A = (y_E \cdot \mathsf{pk}_B^{H(y_E,A)})^a = g^{(x_A+H(y_A\|B)\mathsf{sk}_A)(x_B+H(y_B\|A)\mathsf{sk}_B)}$$

Because Eve knows sk_A, y_A, y_E, she can compute a, however without sk_B, she cannot know the exponent $x_E + H(y_E, A)\mathsf{sk}_B$, provided DLP is hard.

38 Secret Sharing

Assume a group of mutually distrusting parties want to share a secret only when a certain amount of parties collaborate to obtain the secret. Secret sharing allows for such a construction. The idea is to share the secret amongst multiple parties, who use protocols to transform these shares into secrets.

38.1 Threshold Secret Sharing

Let $\mathbf{U} = \{U_1, \ldots, U_n\}$ be a set of n users and let $1 \leq t \leq n$ denote a threshold. Let $A = (t, n)$ denote an access structure that enables any subset of t-out-of-n users to access a shared secret.

A secret sharing scheme $S = (\text{Share}, \text{Recover})$ consists of two protocols:

- $\text{Share}(k, \mathbf{U}, A)$: For each $U_i \in \mathbf{U}$, generate a secret share s_i. Sharing can be probabilistic.
- $\text{Recover}(\{s_i\}_{U_i \in Q}, \mathbf{U}, A)$: If $Q \subseteq \mathbf{U}$ is a qualified subset for A, then use $\{s_i\}_{U_i \in Q}$ to recover k and output it. Recovering is always deterministic.

Correctness $\forall k, \mathbf{U}, A, \{s_1, \ldots, s_n\} := \text{Share}(k, \mathbf{U}, A)$ and any qualified set $Q \subseteq \mathbf{U}$, it holds that

$$\text{Recover}(\{s_i\}_{U_i \in Q}, \mathbf{U}, A) = k$$

38.2 Privacy in Secret Sharing

The intuition behind privacy is that the value k must remain secret for all non-qualified sets of users.

Let λ be the security parameter. The privacy game $\mathsf{Game}^{\text{privacy}}_{\mathcal{A},S}(\lambda)$ is defined as follows.

Fix a set of users U and an access structure A.

1. The dealer D chooses two different secrets k_0 and k_1 such that $|k_0| = |k_1|$.
2. A bit $b \xleftarrow{\$} \{0,1\}$ is chosen uniformly at random.
3. The dealer computes $\{s_1, \ldots, s_n\} := \text{Share}(k_b, U, A)$ and gives share s_i to user U_i.
4. The adversary $\mathcal{A}(k_0, k_1)$ may corrupt any user U_i and receive the corresponding secret share s_i.
5. Finally, $\mathcal{A}$ outputs a bit b'.
6. If $\mathcal{A}$ did not corrupt any qualified set $Q \subseteq U$, then the game returns 1 iff $b' = b$.

The privacy advantage of an adversary is defined by
$\mathsf{Adv}^{\text{privacy}}_S := \left|\Pr\left[\mathsf{Game}^{\text{privacy}}_{\mathcal{A},S}(\lambda) = 1\right] - \frac{1}{2}\right|$.

We distinguish between the following notions of privacy:

Perfect privacy if $\mathsf{Adv}^{\text{privacy}}_S = 0$.

Computational privacy if $\mathsf{Adv}^{\text{privacy}}_S \leq \text{negl}(\lambda)$.

ε-statistical privacy if $\mathsf{Adv}^{\text{privacy}}_S \leq \varepsilon$ for some small constant ε.

Special Case: n-out-of-n Secret Sharing

Let $k \in \{0,1\}^\lambda$ be a secret that a dealer wants to share among users $U = \{U_1, \ldots, U_n\}$. Denote an n-out-of-n access structure A by (n, n).

The sharing and recovery algorithms are defined as follows.

- Share$(k, U, (n, n))$: choose shares $s_2, \ldots, s_n \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random, compute $s_1 := k \oplus s_2 \oplus \cdots \oplus s_n$, and return $\{s_1, \ldots, s_n\}$.
- Recover$(\{s_1, \ldots, s_n\}, U, (n, n))$: return $k := s_1 \oplus \cdots \oplus s_n$.

The above scheme provides perfect privacy. Given any $n-1$ shares, say $\{s_1, \ldots, s_{n-1}\}$, there exist unique values s_{n_0} and s_{n_1} such that, for any two secrets k_0 and k_1,

$$k_0 := s_1 \oplus \cdots \oplus s_{n-1} \oplus s_{n_0} \text{ and } k_1 := s_1 \oplus \cdots \oplus s_{n-1} \oplus s_{n_1}$$

.

Hence, $\Pr\left[\mathsf{Game}^{\text{privacy}}_{\mathcal{A},S}(\lambda) = 1\right] = \frac{1}{2}$, and therefore $\mathsf{Adv}^{\text{privacy}}_S = 0$.

38.3 Lagrange Interpolation of Polynomials

Let $f(x)$ be a polynomial over $\mathbb{R}$ and let (x_i, y_i) denote its points, i.e., $f(x_i) = y_i$. A set of t distinct points $\{(x_1, y_1), \ldots, (x_t, y_t)\}$, where $x_i \neq x_j$ for $i \neq j$, defines the Lagrange polynomial $f(x)$, a unique polynomial of degree at most $t-1$:

$$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$$

The coefficients $\{a_0, \ldots, a_{t-1}\}$ of $f(x)$ can be found using Lagrange interpolation as follows:

$$f(x) = \sum_{i=1}^{t} y_i \cdot \ell_i(x) \text{ with } \ell_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{x_j - x}{x_j - x_i}$$

Example Assume $t = 2$ and we have the points $(1, 3)$ and $(4, 6)$. Then $\ell_1(x) = \frac{4-x}{4-1}$ and $\ell_2(x) = \frac{1-x}{1-4}$ Therefore, $f(x) = 3 \cdot \ell_1(x) + 6 \cdot \ell_2(x) = 2 + x$ Hence, $a_0 = 2$ and $a_1 = 1$.

38.4 Shamir's Secret Sharing

Let $U = U_1, \ldots, U_n$, let $p \gg n+1$ be a large prime number, and let $t \leq n$. Let $k \in \mathbb{Z}_p$ be a secret and let $A = (t, n)$ denote a "t-out-of-n" access structure.

Shamir's secret sharing scheme works as follows:

- Share$(k, U, (t, n))$: Pick coefficients $a_1, \ldots, a_{t-1} \xleftarrow{\$} \mathbb{Z}_p$ and set $a_0 := k$. Define
$$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$$
and return secret shares $s_1, \ldots, s_n$ with $f(i) = s_i$.
- Recover$(s_{i_1}, \ldots, s_{i_t}, U, (t, n))$: Recover and return the secret
$$k = f(0) = \sum_{m=1}^{t} s_{i_m} \cdot \ell_{i_m}(0) \qquad \text{where} \qquad \ell_{i_m} = \prod_{j=1}^{t} \prod_{j \neq m} \frac{i_j}{i_j - i_m}.$$

Example Let $t = 3$, $n = 5$, $p = 17$, and $k = 3$, and let $f(x) = 3 + 14x + 15x^2$ be the dealer-chosen polynomial.

Secret Sharing Phase:
$$\begin{aligned} U_1 &\leftarrow s_1 = f(1) = 15 \bmod 17 = 15, \\ U_2 &\leftarrow s_2 = f(2) = 91 \bmod 17 = 6, \\ U_3 &\leftarrow s_3 = f(3) = 180 \bmod 17 = 10, \\ U_4 &\leftarrow s_4 = f(4) = 299 \bmod 17 = 10, \\ U_5 &\leftarrow s_5 = f(5) = 448 \bmod 17 = 6. \end{aligned}$$

Secret Recovery Phase using s_1, s_3, s_5:
$$\begin{aligned} \ell_1(0) &= \frac{3}{3-1} \cdot \frac{5}{5-1} = 15 \cdot 8^{-1} = 15 \cdot 15 = 4 \pmod{17}, \\ \ell_3(0) &= \frac{1}{1-3} \cdot \frac{5}{5-3} = 5 \cdot (-4)^{-1} = 5 \cdot 4 = 3 \pmod{17}, \\ \ell_5(0) &= \frac{1}{1-5} \cdot \frac{3}{3-5} = 3 \cdot 8^{-1} = 3 \cdot 15 = 11 \pmod{17}. \end{aligned}$$

Then:
$$\begin{aligned} k = f(0) &= s_1 \cdot \ell_1(0) + s_3 \cdot \ell_3(0) + s_5 \cdot \ell_5(0) \\ &= 15 \cdot 4 + 10 \cdot 3 + 6 \cdot 11 \\ &= 156 \bmod 17 \\ &= 3. \end{aligned}$$

Note that for any $a \in \mathbb{Z}_p \setminus 0$, the additive inverse $(-a)$ can be computed as $p - a$. And, for any $a \in \mathbb{Z}_p \setminus 0$, the multiplicative inverse a^{-1} can be computed as $a^{p-2} \bmod p$.

Privacy of Shamir Secret Sharing

Shamir's secret sharing scheme offers perfect privacy.

Proof Assume adversary $\mathcal{A}$ knows $t-1$ shares s_i. There exist $p^{t-(t-1)} = p$ polynomials $f(x) \in \mathbb{Z}_p[x]$ of degree $\leq t-1$ such that $f(i) = s_i$ for all i.

For any $k \in \mathbb{Z}_p$ there exists exactly one polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree $\leq t-1$ with $f(0) = k$ and $f(i) = s_i$ for all i.

Adversary $\mathcal{A}$ can construct all possible $f(x)$ but cannot decide which is the right one. Any $k \in \mathbb{Z}_p$ has the same probability to be chosen as a secret.

Therefore, $\Pr[\mathsf{Game}^{\text{privacy}}_{\mathcal{A},S}(\lambda) = 1] = 1/2$ and $\text{Adv}^{\text{privacy}}_S = 0$.

Misbehavior in Shamir Secret Sharing

Dealer D and users U_i can misbehave in Shamir's secret sharing scheme as follows:

Dealer

- If D chooses a polynomial of degree greater than $t-1$, then any subset of t users will compute different secrets from their shares.
- If D chooses a polynomial of degree smaller than $t-1$, then any subset of $d+1$ users can recover the secret if the chosen polynomial has degree $d < t-1$.
- If D distributes inconsistent, e.g., random, shares to the users, then any subset of t users will compute different secrets from their shares.

Users

- If U_i submits bad share s_i' during the recovery protocol, then honest users will not be able to recover the original secret k.

To detect such misbehavior, secret sharing schemes must offer *verifiability*.

38.5 Verifiable Secret Sharing (VSS)

Assume D gives each U_i a share s_i of secret k such that:

- Any qualified set $Q \subseteq \mathbf{U}$ can reconstruct k.
- Any non-qualified set $Q \subseteq \mathbf{U}$ does not get any information about k.
- Each user U_i can verify validity of its own share s_i through a proof π.

Let $(\mathbb{G}, g, q)$ be a cyclic group of prime order q in which the DL problem is hard. **Feldman's VSS scheme** is as follows:

- Share
  - D chooses $f(x) = a_0 + a_1 x + \ldots + a_{t-1}x^{t-1}$ where $a_0 = k$ and $a_i \in \mathbb{Z}_q$.
  - D computes $A_i = g^{a_i}$ and publishes $(A_0, \ldots, A_{t-1})$ as proof π.
  - D gives share $s_i = f(i)$ to $U_i \forall i$.
  - U_i checks whether $g^{s_i} = \prod_{j=0}^{t-1} A_j^{i^j}$ to verify its share s_i .
- Recover
  - For each submitted share s_i verify that $g^{s_i} = \prod_{j=0}^{t-1} A_j^{i^j}$
  - Use t correct shares to recover k via Lagrange interpolation.

Let $(\mathbb{G}, g, q)$ be a cyclic group of prime order q in which the DL problem is hard and let h be a random generator of $\mathbb{G}$ for which $\log_g h$ is unknown. **Pedersen's VSS scheme** is specified as follows:

- Share
  - D chooses $f(x) = a_0 + a_1 x + \ldots + a_{t-1}x^{t-1}$ where $a_0 = k$ and $a_i \in \mathbb{Z}_q$.
  - D chooses $v(x) = b_0 + b_1 x + \ldots + b_{t-1}x^{t-1}$ where $b_i \in \mathbb{Z}_q$.
  - D computes $A_i = g^{a_i} h^{b_i}$ and publishes $(A_0, \ldots, A_{t-1})$ as proof π.
  - D gives shares $s_i = f(i)$ and $s_i' = v(i)$ to U_i for all i.
  - U_i checks whether $g^{s_i} h^{s_i'} = \prod_{j=0}^{t-1} A_j^{i^j}$ to verify its share s_i.
- Recover
  - For each submitted share (s_i, s_i') verify that $g^{s_i} h^{s_i'} = \prod_{j=0}^{t-1} A_j^{i^j}$.
  - Use t correct shares s_i to recover k via Lagrange interpolation.

While Feldman's VSS scheme is computationally secure (as $A_0 = g^k$ is made public), Pedersen's scheme offers perfect privacy as $A_0 = g^k h^{b_0}$ perfectly hides k.

Advanced Cryptographic Protocols

Introduction to Cryptography (COMP0025)

Dan Ristea | Sem 1 25/26 | 24-28 Nov

---

39 Commitment Protocols

Say we want to conduct an auction, one way of doing this is to have bidders submit their bids sealed, and later opening them and having the highest bid win. There may likely be adversaries who wish to cheat against such a system to bid a higher value than the expected winner, or to bid a lower value to renege on a bid.

Such auctions can be described in two stages, the **bidding stage** should have sealed bid c_i not leak any information about the bid b_i. The **opening stage** should ensure that a bidder opening their bid b_i, cannot open to another bid $b_i' \neq b_i$. The sealed bid auction has analogous properties which are desirable for Commitment Schemes.

A commitment scheme is an interactive protocol between a sender S and a receiver R, consisting of:

- **Commit**: Sender S sends a commitment $c := \mathsf{Commit}(m, r)$ for some message m and randomness r to receiver R.
- **Reveal**: Sender S opens c by sending m and r to receiver R who accepts if $c = \mathsf{Commit}(m, r)$.

A commitment scheme has two properties:

- **Hiding**: R should not be able to obtain any information about m from c.
- **Binding**: S should not be able to open c to any other message $m' \neq m$.

Commitment Scheme Adversaries

- **Hiding-Adversary** $\mathcal{A}$: seeks to learn information about m from $c := \mathsf{Commit}(m, r)$, playing the role of the receiver R.
- **Binding-Adversary** $\mathcal{A}$: seeks to find (m, r) and (m', r') with $\mathsf{Commit}(m, r) = \mathsf{Commit}(m', r')$ and $m \neq m'$, playing the role of the sender S.

39.1 Pedersen Commitments

Let $\mathbb{G}$ be a cyclic group of order q and generator g in which the DLP is hard. Let h be a randomly chosen generator of $\mathbb{G}$ with $h \neq g$ and assume the discrete logarithm x such that $h = g^x$ is unknown.

The Pedersen commitment scheme is:

- **Commit**: Sender S chooses $m \in \mathbb{Z}_q$, picks $r \xleftarrow{\$} \mathbb{Z}_q$, and sends $c := g^m h^r$ to receiver R.
- **Reveal**: Sender S opens c by sending m and r to receiver R who accepts if $c = g^m h^r$.

Pedersen Commitments: Hiding

To prove the Pedersen commitment scheme is perfectly hiding, we show that for any $m' \neq m$, there exists an r' such that $c = g^{m'} h^{r'}$.

Assume $x = \log_g h$ is known to an information-theoretic adversary $\mathcal{A}$. For any $c = g^m h^r$ returned in the commitment stage there exists an a with $c = g^a$. Since $r \xleftarrow{\$} \mathbb{Z}_q$, the value $a = m + xr$ is uniformly distributed in $\mathbb{Z}_q$. For any given message $m' \in \mathbb{Z}_q$ there exists a unique $r' \in \mathbb{Z}_q$ such that $c = g^{m'} h^{r'}$.

Pedersen Commitments: Binding

Assuming the DLP is hard, the Pedersen commitment scheme is computationally binding.
First assume that the Pedersen commitment scheme does not provide computational binding, and let $\mathcal{A}$ be a PPT adversary which breaks the binding property. Let $\mathcal{B}$ be a PPT adversary against the DLP that simulates Pedersen commitments for $\mathcal{A}$. We reduce binding to DLP as follows:

- $\mathcal{A}$ outputs two openings (m, r) and (m', r') for the same commitment with $m \neq m'$.
- If $\mathcal{A}$ breaks binding, then $g^m h^r = g^{m'} h^{r'}$ and $m + xr \equiv m' + xr' \pmod{q}$. Thus $x \equiv (m - m') \cdot (r' - r)^{-1} \pmod{q}$ and $\mathcal{B}$ breaks the DLP since $g^x = h$.

The DLP game:

$\mathsf{Game}^{\mathrm{DLP}}_{\mathcal{B},\mathrm{Gen}}(\lambda)$:
$(\mathbb{G}, g, q) \leftarrow \mathrm{Gen}(\lambda)$
$h \xleftarrow{\$} \mathbb{G}$
$x \leftarrow \mathcal{B}(\mathbb{G}, g, q, h)$
if $g^x = h$ **then**
  return 1
end
return 0

39.2 Commitment Schemes from CRHFs

Let H be a CRHF accepting arbitrary size inputs. A hash-based commitment scheme is specified as follows:

- **Commit**: Sender S chooses $m \in \{0,1\}$, picks $r \xleftarrow{\$} \{0,1\}^\lambda$, and sends $c := H(m \| r)$ to receiver R.
- **Reveal**: Sender S opens c by sending m and r to receiver R who accepts if $c = H(m \| r)$.

The CRHF commitment scheme with H in the ROM is hiding as an adversary queries $H(* \| r)$ only with negligible probability and is binding since H is collision-resistant. Note: the randomness length λ must be large enough to make brute-force attacks infeasible. Merkle trees also allow to commit efficiently to large sets of values and are very often used in practice (in blockchain systems).

40 Identification Protocols

Imagine a scenario in which a party A wants to authenticate to another party B. For example, A could be a user who wants to authenticate to a web server B. Assume A has a key pair $(\mathsf{sk}, \mathsf{pk})$ and B knows pk, then A can authenticate to B using the following challenge-response identification scheme (IDS):

- **Challenge**: B sends a random challenge c to A.
- **Response**: A computes a response r as a function of c and sk and sends it to B.
- **Verification**: B accepts A if r is valid with respect to c and pk and rejects otherwise.

IDS protocols can be realized, e.g., through digital signatures or public-key encryption.

Signature-based Challenge-Response Identification

Let $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme. Assume A has $(\mathsf{sk}, \mathsf{pk}) := \mathsf{Gen}(\lambda)$ and B knows pk. Then A can authenticate to B as follows:

- B sends a random challenge $c \xleftarrow{\$} \{0,1\}^\lambda$ to A.
- A replies with signature $\sigma := \mathsf{Sign}(\mathsf{sk}, c)$.
- B accepts A if $\mathsf{Verify}(\mathsf{pk}, c, \sigma) = 1$ and rejects otherwise.

If DS is EUF-CMA secure, then DS-IDS is an identification scheme that is secure against impersonation attacks by active adversaries.

Proof By a reduction to EUF-CMA of DS. Assume there exists a PPT adversary $\mathcal{A}$ who impersonates A in the DS-IDS scheme with non-negligible probability. We construct a PPT adversary $\mathcal{F}$ against EUF-CMA of the signature scheme.

$\mathcal{F}$ is given pk and access to a signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$. Their goal is to output (m', σ') such that $\mathsf{Verify}(\mathsf{pk}, m', \sigma') = 1$ and m' was never queried to the signing oracle.

$\mathcal{F}$ runs $\mathcal{A}$ and simulates the identification environment for it. Since $\mathcal{A}$ is active, it may interact with honest instances of A before attempting to impersonate them. Whenever $\mathcal{A}$ sends a challenge c to an honest prover, $\mathcal{F}$ answers by querying its signing oracle on c and returning $\sigma := \mathsf{Sign}(\mathsf{sk}, c)$ to $\mathcal{A}$. Thus, $\mathcal{A}$ views the simulation is perfect as every transcript it sees has the same distribution as in the real identification scheme.

$\mathcal{A}$ eventually impersonates A to an honest verifier B. $\mathcal{F}$ plays the role of B and samples a fresh random challenge $c' \xleftarrow{\$} \{0,1\}^\lambda$. Sends c' to $\mathcal{A}$. If $\mathcal{A}$ succeeds, it returns σ' such that $\mathsf{Verify}(\mathsf{pk}, c', \sigma') = 1$. The reduction $\mathcal{F}$ then outputs (c', σ') as its EUF-CMA forgery.

Suppose $\mathcal{A}$ made at most $q(\lambda)$ signing queries, for some polynomial q. Since c' was sampled uniformly from $\{0,1\}^\lambda$ independently of previous interactions, the probability that c' appears in a previous query is at most $\frac{q(\lambda)}{2^\lambda} \leq \mathrm{negl}(\lambda)$.

Then, whenever $\mathcal{A}$ successfully impersonates A on the challenge c', $\mathcal{F}$ obtains a valid signature on a new message. Thus

$$\Pr[\mathcal{F} \text{ wins EUF-CMA}] \geq \Pr[\mathcal{A} \text{ successfully impersonates } A] - \frac{q(\lambda)}{2^\lambda}$$

If $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{F}$ also wins the EUF-CMA game with non-negligible probability, contradicting our original assumption. Therefore, no probabilistic polynomial time adversary can impersonate A with non-negligible probability.

PKE-based Challenge-Response Identification

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Assume A has $(\mathsf{sk}, \mathsf{pk}) := \mathsf{Gen}(\lambda)$ and B knows pk. Then A can authenticate to B as follows:

- B picks a random message $m \xleftarrow{\$} \{0,1\}^n$ and sends $c := \mathsf{Enc}(\mathsf{pk}, m)$ as a challenge to A.
- A decrypts $m' := \mathsf{Dec}(\mathsf{sk}, c)$ and replies with m'.
- B accepts A if $m' = m$ and rejects otherwise.

If PKE is IND-CCA secure, then PKE-IDS is an identification scheme that is secure against impersonation attacks by active adversaries.

Proof by a reduction to IND-CCA of PKE. Assume there exists a PPT adversary $\mathcal{A}$ who impersonates A in the PKE-IDS scheme with non-negligible probability. We construct a PPT adversary $\mathcal{D}$ against IND-CCA of PKE.

$\mathcal{D}$ is given pk and access to a decryption oracle $\mathsf{Dec}(\mathsf{sk}, \cdot)$. To win, $\mathcal{D}$ must distinguish an encryption of one of two messages of its choice.

$\mathcal{D}$ runs $\mathcal{A}$ and simulates the identification environment for it. Since $\mathcal{A}$ is active, it can interact with honest instances of A before attempting an impersonation to an honest verifier. Whenever $\mathcal{A}$ sends some c as a challenge to an honest prover instance, $\mathcal{D}$ answers by querying its decryption oracle and returning $m := \mathsf{Dec}(\mathsf{sk}, c)$ to $\mathcal{A}$, thus, all honest prover interactions are simulated perfectly.

Eventually, $\mathcal{A}$ attempts to impersonate A to an honest verifier, to which $\mathcal{D}$ can now choose two distinct messages $m_0, m_1 \xleftarrow{\$} \{0,1\}^n$ and submits (m_0, m_1) to its IND-CCA challenger. The challenger samples $b \xleftarrow{\$} \{0,1\}$ and returns the challenge ciphertext $c' := \mathsf{Enc}(\mathsf{pk}, m_b)$.

If $\mathcal{A}$ successfully impersonates A, it must return some $m' = m_b$. The reduction $\mathcal{D}$ then outputs its guess as

$$\hat{b} = \begin{cases} 0, & \text{if } m' = m_0, \\ 1, & \text{if } m' = m_1, \\ \text{a uniform bit}, & \text{otherwise.} \end{cases}$$

$\mathcal{D}$'s success probability is strictly greater than the probability of $\mathcal{A}$ succeeding in its outputting of a correct plaintext m_b, hence

$$\Pr[\mathcal{D} \text{ wins IND-CCA}] \geq \epsilon(\lambda) + \frac{1 - \epsilon(\lambda)}{2} = \frac{1}{2} + \frac{\epsilon(\lambda)}{2}$$

where

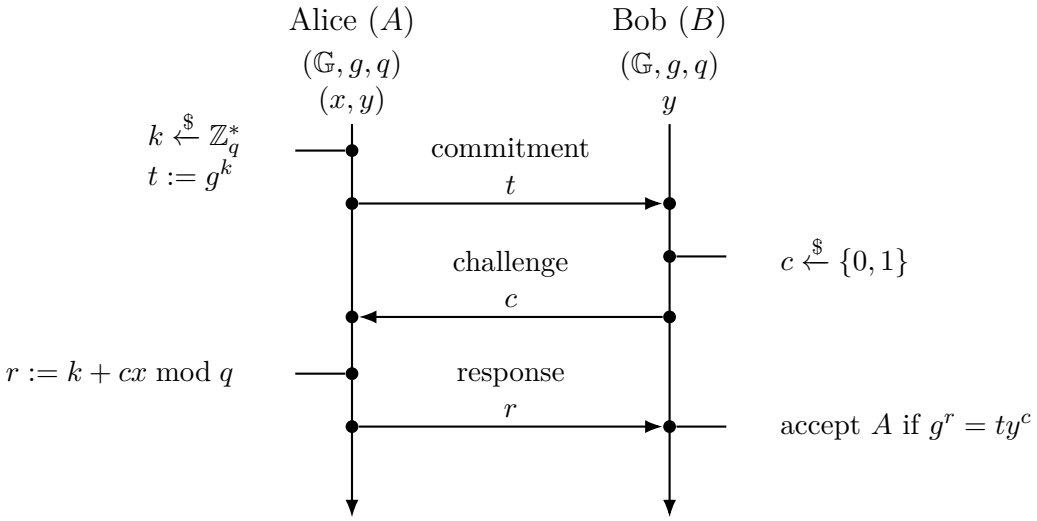
$$\epsilon(\lambda) = \Pr[\mathcal{A} \text{ successfully impersonates } A]$$

Therefore, if $\mathcal{A}$ impersonates A with non-negligible probability, then $\mathcal{D}$ breaks IND-CCA security of PKE with a non-negligible advantage, contradicting our original assumption.

40.1 Schnorr Identification Scheme

Let $\mathbb{G}$ be a group of order q with generator g in which the DLP is hard, e.g., $\mathbb{G} \subseteq \mathbb{Z}_p^*$ of order q for an appropriate prime p. Let A and B be two users where A holds private key $x \in \mathbb{Z}_q^*$ and public key $y = g^x$

and B holds y. The Schnorr identification system works as follows.



Note that $g^r = g^k g^{cx} = ty^c$.

Can $\mathcal{A}$ impersonate A without knowing x? Yes, with at least $\frac{1}{2}$ probability if $\mathcal{A}$ does not abort in the following attack:

- $\mathcal{A}$ picks $c' \xleftarrow{\$} \{0,1\}$ and $r \in \mathbb{Z}_q^*$ and sends the commitment $t = g^r y^{-c'}$ to B.
- B replies with $c \xleftarrow{\$} \{0,1\}$
- $\mathcal{A}$ sends response r to B if $c = c'$ and aborts otherwise.
- B accepts A (alias $\mathcal{A}$) if $g^r = ty^c$.

If B asks A to repeat the protocol λ times without aborting in any of them then $\mathcal{A}$ can succeed with probability at least $2^{-\lambda}$ which is negligible in λ.

Can an adversary $\mathcal{A}$ increase its probability to win without knowing x? Yes, but only if $\mathcal{A}$ is able to send a valid response r for both cases $c = 0$ and $c = 1$.

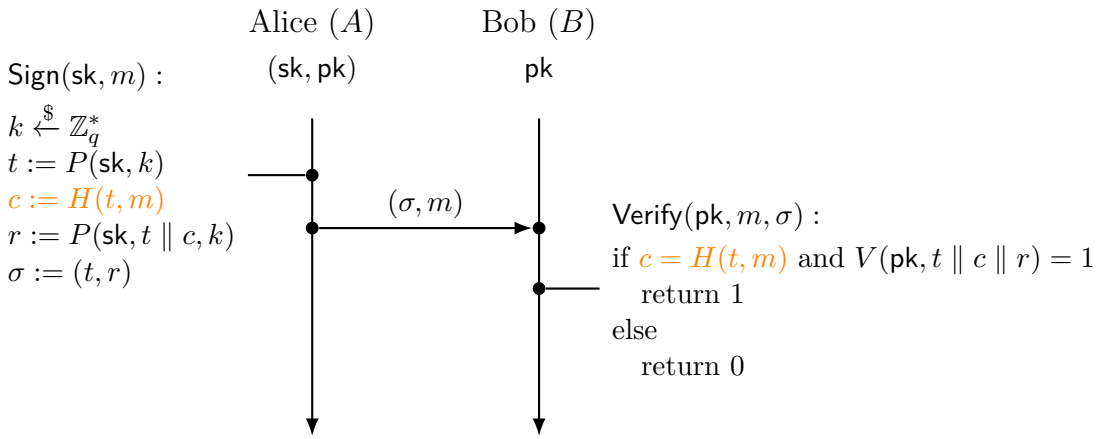
- Let $c_0 \xleftarrow{\$} \{0,1\}$ and $c_1 = 1 - c_0$ be the possible challenges sent by B.
- Since B would accept in both cases we have $t = g^{r_0} y^{-c_0} = g^{r_1} y^{-c_1}$,i.e.,$g^{r_0 - r_1} = y^{c_0 - c_1}$ where $(c_0 - c_1) = \pm 1$.
- Since $(r_0 - r_1) = x \cdot (c_0 - c_1) \bmod q$, we can extract $x = (r_0 - r_1) \cdot (c_0 - c_1)^{-1} \bmod q$.
- This is the special-soundness property: from two accepting transcripts with the same commitment and different challenges, an extractor can compute x.

40.2 Fiat–Shamir transformation

In the IDS A computes response r on an unpredictable challenge c chosen by B. Note that A receives c after it sent its commitment t.

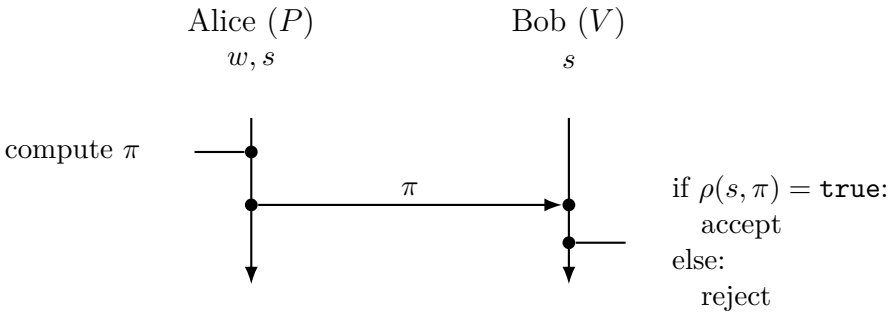
The Fiat–Shamir transformation converts a canonical IDS into a digital signature scheme $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ by having A compute c based on t without interacting with B in such a way that A cannot change t later.

Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be modeled as a random oracle and let $m \in \{0,1\}^\lambda$ be a message to be signed.



41 Zero-Knowledge Proofs

Let's imagine Alice wants to prove to Bob she has knowledge over some statement, without revealing anything else.



An Interactive Proof System (IPS) is a protocol between prover P and verifier V in which:

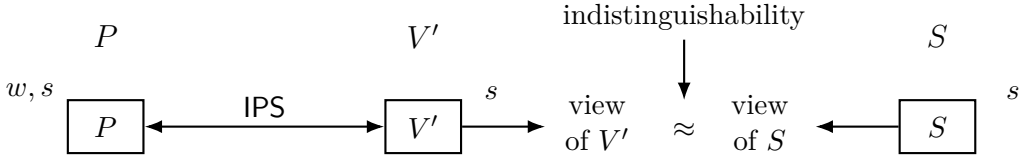
- P tries to convince V to accept the validity of some statement s.
- P has a secret w to attest validity of s and needs to come up with a proof π.
- V can verify the validity of s using some publicly verifiable relation ρ such that

$$\rho(s, \pi) = \begin{cases} 1, & \text{if } \pi \text{ is a valid proof for } s \\ 0 & \text{otherwise} \end{cases}$$

Properties of Interactive Proof Systems

Completeness - If a given statement s is true and an untrusted prover P has a valid secret w for s then verifier V always accepts P's proof π, i.e., $\Pr[\rho(s, \pi) = 1] = 1$.
Soundness - If a given statement s is false then verifier V rejects P's proof π except for some negligible probability $\mathrm{negl}(\lambda)$ in the security parameter λ, i.e., $\Pr[\rho(s, \pi) = 1] \leq \mathrm{negl}(\lambda)$. This holds even if P misbehaves and does not follow the protocol.

41.1 Zero-Knowledge Proofs



IPS soundness protects an honest verifier V against a malicious prover P'. However, can we also protect an honest prover P against a malicious verifier V' who tries to extract the secret w from a valid statement-proof pair (s, π)?

Zero Knowledge aims to protect the secrecy of w and enables P to prove validity of s via π without revealing any information about w. Defined through the indistinguishability of views output by V' and a simulator S.

Proving knowledge of Private Keys

Assume a user P wants to obtain a certificate for y from a certificate authority. How can P, who generated (x, y), prove knowledge of x to the CA in zero-knowledge?

Let $\mathbb{G}$ be a cyclic group of order q with generator g in which the DLP is hard. Let $x \in \mathbb{Z}_q^*$ be a private key and $y = g^x \in \mathbb{G}$ be the corresponding public key.

We want to prove the following statement "*P knows the discrete logarithm of y to base g*" So, P just uses the private key $x = \log_g y$ to prove this. P runs the following:

- P picks $k \xleftarrow{\$} \mathbb{Z}_q$ and sends commitment $t := g^k$ to V.
- V replies with a random challenge $c \xleftarrow{\$} 0, 1$ to P.
- P then sends response $r := k + cx \bmod q$ to V.
- V accepts if $g^r = ty^c$ and rejects otherwise.

Such a proof is denoted $\mathsf{ZKPoK}[x|y = g^x]$, this protocol is identical to Schnorr's IDS.

$\mathsf{ZKPoK}[x|y = g^x]$ is zero-knowledge because a PPT simulator S can output a view indistinguishable from the view of V'. V's view of the protocol is (t, c, r) such that $g^r = ty^c$.

A simulator can pick $r \xleftarrow{\$} \mathbb{Z}_q, c \xleftarrow{\$} \{0, 1\}$ and compute $t = g^r y^{-c}$ and output (t, c, r). For λ repetitions, indistinguishability holds only if V' chooses c at random. Therefore, $\mathsf{ZKPoK}[x|y = g^x]$ is an *honest-verifier* ZKP system.

Zero-Knowledge Proofs and Signatures

Assume P, who knows $(sk, pk) := (x, y)$, authenticates to V with a signature. Is the following challenge-response protocol zero-knowledge?

- V picks $c \xleftarrow{\$} \{0, 1\}$ and sends it to P.
- P replies with signature $\sigma := \mathsf{Sign}(\mathsf{sk}, c)$ to V.
- V accepts if $\mathsf{Verify}(\mathsf{pk}, c, \sigma) = 1$ and rejects otherwise.

The view of V' is (c, σ) such that $\mathsf{Verify}(\mathsf{pk}, c, \sigma) = 1$. A simulator S, given pk, must output an indistinguishable view (c, σ). However, S then must forge a signature, therefore this protocol is not ZK.

Zero-Knowledge Proofs of Discrete Log Equality

If prover P wants to prove to V the equality of the discrete logarithms of y_1 with respect to base g and of y_2 with respect to base h, i.e., $\mathsf{ZKPoK}[x|y_1 = g^x \wedge y_2 = h^x]$, they can run the following protocol:

- Commitment: P picks $k \xleftarrow{\$} \mathbb{Z}_q$ and sends $t_1 = g^k$ and $t_2 = h^k$ to V.
- Challenge: V replies with a random challenge $c \xleftarrow{\$} \{0, 1\}$ to P.
- Response: P then replies with $r := k + cx \bmod q$ to V.
- Verification: V accepts if $g^r = t_1 y_1^c$ and $h^r = t_2 y_2^c$ and rejects otherwise.

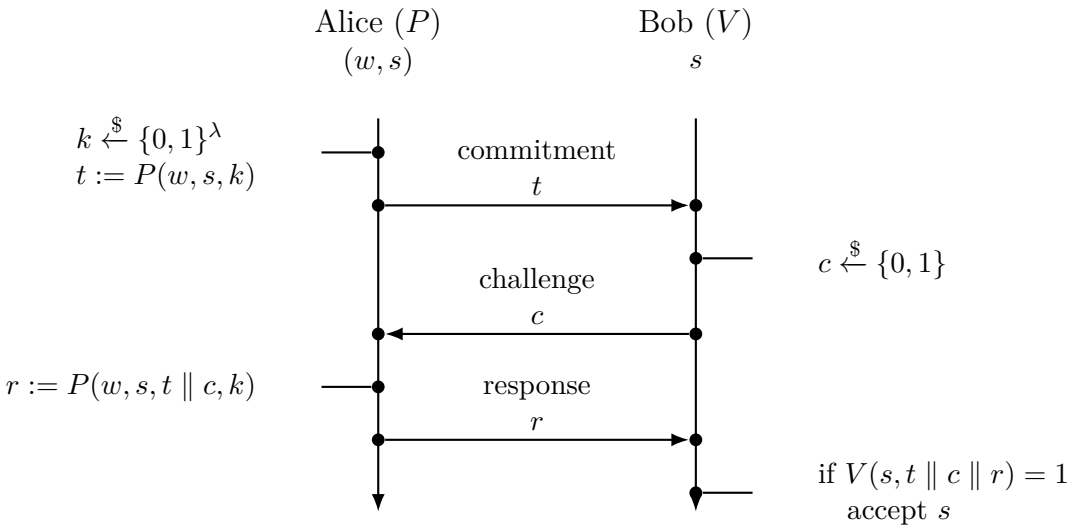
Note that; $g^r = g^{k+cx} = g^k \cdot (g^x)^c = t_1 \cdot y_1^c$ and $h^r = h^{k+cx} = h^k \cdot (h^x)^c = t_2 \cdot y_2^c$. Repeating the above protocol λ times, negates a malicious prover's probability of success to $2^{-\lambda}$ which is negligible in λ.

The view of V' in $\mathsf{ZKPoK}[x|y_1 = g^x \wedge y_2 = h^x]$ is (t_1, t_2, c, r) such that $g^r = t_1 \cdot y_1^c$ and $h^r = t_2 \cdot y_2^c$.

A simulator S on input of y_1, y_2 picks $r \xleftarrow{\$} \mathbb{Z}_q, c \xleftarrow{\$} \{0, 1\}$, computes $t_1 = g^r y_1^{-c}$ and $t_2 = h^r y_2^{-c}$ and outputs (t_1, t_2, c, r) For λ repetitions, indistinguishability holds only if V' chooses c at random. Therefore, $\mathsf{ZKPoK}[x|y_1 = g^x \wedge y_2 = h^x]$ is an *honest-verifier* ZKP system.

Relationship with Canonical IDS

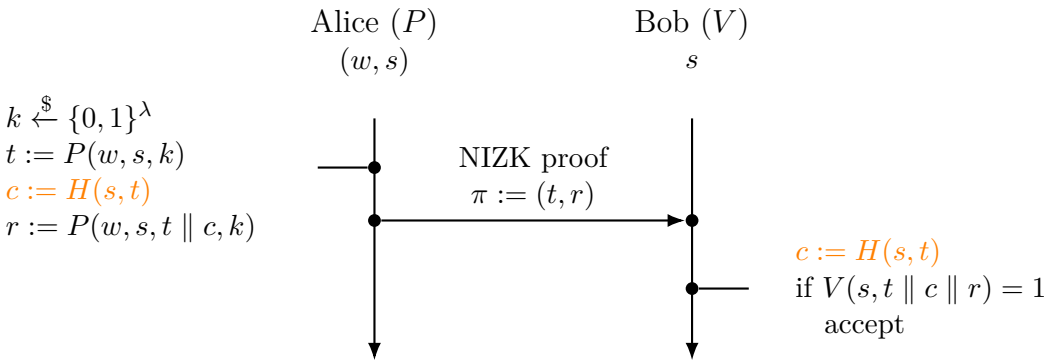
ZKPs are interactive and typically follow the same pattern as canonical IDS.



Canonical IDS can be converted into digital signatures using the Fiat-Shamir transformation. But can we remove the interaction with V and create a non-interactive ZKP?

41.2 Non-Interactive Zero-Knowledge (NIZK) Proofs

Yes, using the same Fiat–Shamir trick as canonical IDS. We let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be modeled as a random oracle; collision resistance alone is not the assumption used by Fiat–Shamir analyses. Then:



Notice the length of $|c|$ is now λ-bits, and the probability of guessing a λ-bit challenge is $2^{-\lambda}$ which is equivalent to the probability of guessing λ single-bit c in interactive HVZKPs. A non-interactive zero-knowledge proof-of-knowledge of the discrete logarithm of y to base g (notation $\mathsf{ZKPoK}[x|y = g^x]$) can be computed as follows:

- P picks $k \xleftarrow{\$} \mathbb{Z}_q$, computes $t = g^k$, $c = H(\mathbb{G}, g, q, y, t)$, and $r = k + cx \bmod q$, then sends $\pi := (t, r)$ to V.
- V computes $c = H(\mathbb{G}, g, q, y, t)$ and accepts if $g^r = ty^c$.

This is similar to Schnorr signatures which have:

- $\mathsf{Sign}(x, m)$: Pick $k \xleftarrow{\$} \mathbb{Z}_q$, compute $t = g^k$, $c = H(t, m)$, and $r = k + cx \bmod q$, then return $\sigma := (c, r)$.
- $\mathsf{Verify}(y, m, \sigma)$: Return 1 if $c = H(g^r \cdot y^{-c}, m)$ and 0 otherwise.

NIZK proofs obtained via the Fiat-Shamir transformation can also use $\pi := (c, r)$. Fiat–Shamir applies to suitable public-coin, sigma-style protocols under additional assumptions, typically in the random-oracle model; it is not a generic transformation for arbitrary ZKPs.