(1)

Claim: H is not collision-resistant

Proof: Since H has the property that $\forall x, y \in \{0,1\}^{2n}$, $(x \leq y) \Rightarrow (H(x) \leq H(y))$, with $H: \{0,1\}^{2n} \to \{0,1\}^{n}$, we know the set of all possible $H(x) \in \{0,1\}^{n}$ is sorted in increasing order when we place each $H(x)$ at its respective index, x.
(This looks like this: $H(x)$: $H(x^{(0)}) \leq H(x^{(1)}) \leq H(x^{(2)}) \leq \cdots \leq \cdots$
x: $x^{(0)} \leq x^{(1)} \leq x^{(2)} \leq \cdots \leq \cdots$)

Here's an algorithm:

```
A(x):
    Compute H(x);
    Set the following variables:
    begin = x+1;
    end = max({0,1}^{2n});
    while (begin ≤ end):
        y = L (end-begin)/2 ⌋ + begin;
        if H(x) = H(y):
            return (x, y)
        else: (means H(x) < H(y))
            end = y;
            repeat while loop;
    if loop ends w/out collisions, return NO COLLISIONS;
```

The Algorithm A above successfully finds a collision on x, y if the collision exists, in, at most, $O(\log(n))$ (since it implements a binary search approach.

We have 2 cases, given some $x \in \{0,1\}^{2n}$, we will show that x has a collision with very high probability:

Case 1: x has a collision

$$\mathbb{P}[\text{Case 1}] = \mathbb{P}[x \in \{0,1\}^{2n} \text{ has a collision}]$$

$$= \frac{\#\text{ of distinct } x \in \{0,1\}^{2n} \text{ w/ collisions}}{\#\text{ of all possible } 2n\text{-bit binary strings}}$$

$$= \frac{\#\text{ of collisions}}{2^{2n}}$$

We know $\#$ of collisions $\geq 2^{2n} - 2^n$, since there are 2^n possible $H(x)$ values, and 2^{2n} possible x values (assign each of the 2^n possible x values a distinct $H(x)$; now the rest of x values will have an $H(x)$ that we've already calculated).

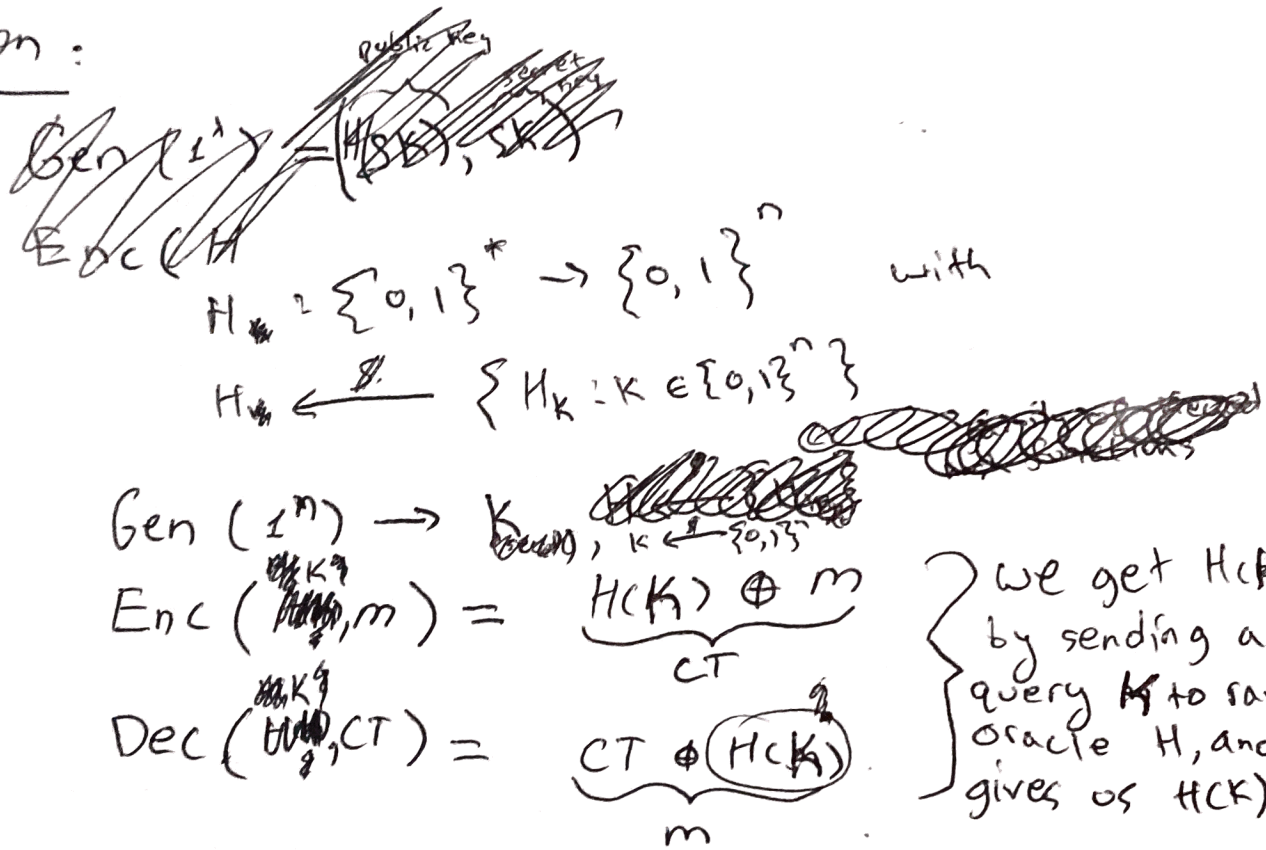
$$\Rightarrow \mathbb{P}[\text{Case 1}] = \frac{2^{2n} - 2^n}{2^{2n}} = \frac{2^n(2^n - 1)}{2^{2n}} = \boxed{\frac{2^n - 1}{2^n}}$$

So, already, we can see $A(x)$ finds a collision (x, y) with non-negligible probability. □

② $H: \{0,1\}^* \to \{0,1\}^n$

Task: Construct a multi-message secure encryption scheme in the random oracle model.

Solution:



Because we use xor operation, correctness of (Gen, Enc, Dec) is evident.

=> Does it satisfy multi-message security?

Need to show

$$D_0 = \{Enc(k, m_{00}), Enc(k, m_{01}), \dots, Enc(k, m_{0q})\}$$

$$\approx_c$$

$$D_1 = \{Enc(k, m_{10}), Enc(k, m_{11}), \dots, Enc(k, m_{1q})\}$$

Proof

First, we inspect a single pair of messages,

$$\{Enc(m_0^{(0)}, K)\} \;\&\; \{Enc(m_1^{(0)}, K)\}$$

$\Rightarrow$ Want to show $\{H(K) \oplus m_0^{(0)}\} \approx_c \{H(K) \oplus m_1^{(0)}\}$

We construct the following Hybrids:

$$H_1 := \{ct := H(K) \oplus m_0^{(0)} \; ; \; K \xleftarrow[(K \xleftarrow{\$} \{0,1\}^n)]{} Gen(1^n)\}$$

$$H_2 := \{ct := r \oplus m_0^{(0)} \; ; \; r \xleftarrow{\$} \{0,1\}^n\}$$

$$H_3 := \{ct := r \oplus m_1^{(0)} \; ; \; r \xleftarrow{\$} \{0,1\}^n\}$$

$$H_4 := \{ct := H(K) \oplus m_1^{(0)} \; ; \; K \xleftarrow[(K \xleftarrow{\$} \{0,1\}^n)]{Gen} (1^n)\}$$

Via the Random Oracle model, $\{H(K); K \xleftarrow{\$} \{0,1\}^n\} \approx_c \{r; r \xleftarrow{\$} \{0,1\}^n$

By closure property then, $H_1 \approx_c H_2$ and $H_3 \approx_c H_4$.

By one-time perfect security of OTP, $H_2 \equiv H_3$,

$$\Rightarrow H_1 \approx_c H_2 \equiv H_3 \approx_c H_4 \Rightarrow \{Enc(m_0^{(0)}, K)\} \approx_c \{Enc(m_1^{(0)}, K)\}$$

Call this result (I)

Now, we can construct the following Hybrid functions:

$$H^{(0)} = \{Enc(k, m_0^{(0)}), Enc(k, m_0^{(1)}), \ldots, Enc(k, m_0^{(q)})\}$$

$$H^{(1)} = \{Enc(k, m_1^{(0)}), Enc(k, m_0^{(1)}), \ldots, Enc(k, m_0^{(q)})\}$$

$$H^{(2)} = \{Enc(k, m_1^{(0)}), Enc(k, m_1^{(1)}), Enc(k, m_0^{(2)}), \ldots, Enc(k, m_0^{(q)})\}$$

$$\vdots$$

$$H^{(q+1)} = \{Enc(k, m_1^{(0)}), Enc(k, m_1^{(1)}), \ldots, Enc(k, m_1^{(q)})\}$$

By what we have shown in (I), we know

$H^{(0)} \approx_c H^{(1)}$, $H^{(1)} \approx_c H^{(2)}$, ..., and $H^{(q)} \approx_c H^{(q+1)}$

By Hybrid Lemma, $\Rightarrow H^{(0)} \approx_c H^{(q+1)}$

$$\Rightarrow D_0 \approx_c D_1$$

□

(5) $\underbrace{H_1: \{0,1\}^{4n} \to \{0,1\}^{2n} \ \& \ H_2: \{0,1\}^{2n} \to \{0,1\}^{n}}_{\text{collision-resistant functions.}}$, hash keys of K_1 & K_2 respectively

Define $H: \{0,1\}^{4n} \to \{0,1\}^{n}$ as such.

$$H(x) := H_2(H_1(x)) \ \forall \ x \in \{0,1\}^{4n}, \text{ w/}$$

the hash key of H being $K = (K_1 \| K_2)$

Claim: H is collision-resistant.

Proof: Suppose not; $\Rightarrow \exists$ PPT $\mathcal{A}$ that can find a collision (x,y) with $x \neq y$ & $H(x) = H(y)$ with nonnegligible probability $\varepsilon(n)$.

$\Rightarrow \mathbb{P}[\mathcal{A} \text{ can find } (x,y),\ x \neq y,\ H_2(H_1(x)) = H_2(H_1(y))] = \varepsilon(n)$

But, note that, given a collision (x,y), we can immediately give two values, $x' = H_1(x)$ and $y' = H_1(y)$. Now there's two possible cases: one where $x' = y'$, and one where $x' \neq y'$.

- Case 1: $x' = y' \Rightarrow H_1(x) = H_1(y)$ and $x \neq y \Rightarrow (x \neq y)$ is a collision on H_1
- Case 2: $x' \neq y' \Rightarrow H_2(x') = H_2(y')$ thus (x', y') is a collision on H_2

$\Rightarrow$ given H is not collision-resistant, either H_1 or H_2 must be not collision resistant $\Rightarrow$ a contradiction $\Rightarrow$ H is collision-resistant.

(4)

Claim: (CPA-secure public-key encrgption scheme)

$\Rightarrow$

(multi-message secure private-key encryption scheme)

Proof: We have the following:

(Gen, Enc, Dec) such that

$Gen(1^\lambda) \rightarrow (pk, sk)$

$Enc(pk, m) \rightarrow CT$

$Dec(sk, CT) = m$

We want to construct (Gen', Enc', Dec') that satisfies private-key multi-message security.

Consider the following scheme:

$Gen'(1^\lambda) = K = (pk \| sk)$, where K is the private key

$Enc'(K, m) = Enc'((pk \| sk), m) = Enc(pk, m) = CT$

$Dec'(K, CT) = Dec'((pk \| sk), CT) = Dec(sk, CT) = m$

Now, we need to show that (Gen', Enc', Dec') satisfies multi-message security; i.e. that is,

define $D_0 = \{Enc'(K, m_0^{(0)}), Enc'(K, m_0^{(1)}), \ldots, Enc'(K, m_0^{(q)})\}$

and $D_1 = \{Enc'(K, m_1^{(0)}), Enc'(K, m_1^{(1)}), \ldots, Enc'(K, m_1^{(q)})\}$.

We want to show $D_0 \approx_c D_1$.

Define the following Hybrids:

$D_0 \to H_0 := \{Enc'(k, m_0^{(0)}), Enc'(k, m_0^{(1)}), \dots, Enc'(k, m_0^{(q)})\}$

Hybrid Lemma tree:

$H_1 := \{Enc(pk, m_0^{(0)}), Enc(pk, m_0^{(1)}), \dots, Enc(pk, m_0^{(q)})\}$

$H_2 := \{Enc(pk, m_1^{(0)}), Enc(pk, m_0^{(1)}), \dots, Enc(pk, m_0^{(q)})\}$

$H_3 := \{Enc(pk, m_1^{(0)}), Enc(pk, m_1^{(1)}), \dots, Enc(pk, m_0^{(q)})\}$

$\vdots$

$H_{q+1} := \{Enc(pk, m_1^{(0)}), Enc(pk, m_1^{(1)}), \dots, Enc(pk, m_1^{(q)})\}$

$D_1 \to H_{q+2} := \{Enc'(k, m_1^{(0)}), Enc'(k, m_1^{(1)}), \dots, Enc'(k, m_1^{(q)})\}$

(I) By the definition we gave, $H_0 \equiv H_1$.

(II) By CPA-security of (Gen, Enc, Dec), $H_i \approx_c H_{i+1} \quad \forall i \in \{1, \dots, q\}$

(III) By Hybrid Lemma, $H_1 \approx_c H_{q+1}$

(IV) Because each pk comes with a unique sk, i.e. the correctness of (Gen, Enc, Dec), we ~~know~~ can conclude $H_{q+1} \equiv H_{q+2}$

$$\Rightarrow H_0 \equiv H_1 \approx_c H_2 \approx_c \cdots \approx_c H_{q+1} \equiv H_{q+2}$$

By Hybrid Lemma $\Rightarrow$ $H_0 \approx_c H_{q+2}$

$$\Rightarrow D_0 \approx_c D_1$$

$\Rightarrow$ (Gen', Enc', Dec') is multi-message secure. □

(5)

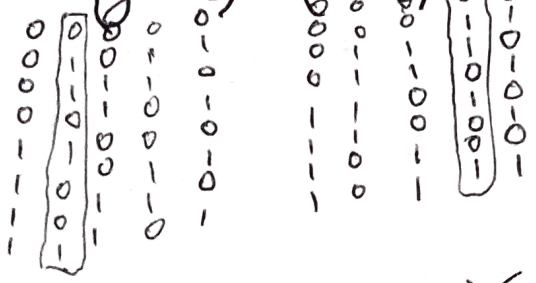
(I) $(\mathbb{Z}_2, \oplus)$ is a group, where $\mathbb{Z}_2 = \{0, 1\}$ & $\oplus$ is XOR

~~$(\mathbb{Z}_2, \oplus)$~~

✓ (i) Associativity: $x, y, z \in \{0, 1\}$

Is $x \oplus (y \oplus z) = (x \oplus y) \oplus z$? Yes.

Truth table:



✓ (ii) Invertibility: Yes.

Case 1: $x = 0 \Rightarrow$ set $y = 1 \Rightarrow x \oplus y = y \oplus x$

Case 2: $x = 1 \Rightarrow$ set $y = 0 \Rightarrow x \oplus y = y \oplus x$

✓ (iii) Identity: Yes.

$(\mathbb{Z}_2, \oplus)$ has an identity. $I = 0$

$\forall x \in \{0, 1\} = \mathbb{Z}_2$, $x \oplus I = I \oplus x$

✓ (iv) Closure: $\forall x, y \in \{0, 1\}$, $x \oplus y \in \{0, 1\} \Rightarrow$ Yes.

$\Rightarrow$ $(\mathbb{Z}_2, \oplus)$ is a group. □

(II) Is $(\mathbb{Z}_p^*, \times)$ a group?

Here, $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$

and $\forall x, y \in \mathbb{Z}_p^*$, $x * y = (xy) \bmod p$

✓ (i) Associativity – Yes

$\forall x, y, z \in \mathbb{Z}_p^*$, $x * (y * z) \stackrel{?}{=} (x * y) * z$

$\Rightarrow x * (yz \bmod p) \stackrel{?}{=} (xy \bmod p) * z$

$\Rightarrow (x(yz \bmod p)) \bmod p \stackrel{?}{=} ((xy \bmod p)z) \bmod p$

Both are $= (xyz) \bmod p = ((x \bmod p)(y \bmod p)(z \bmod p)) \bmod p$,

so $(\mathbb{Z}_p^*, *)$ possesses associativity.

✓ (ii) Invertibility: – Yes

$\forall x \in \mathbb{Z}_p^*$, $\exists y$ s.t. $x * y = y * x$

$\Rightarrow \underbrace{(xy) \bmod p = (yx) \bmod p}_{\text{true } \forall x, y \in \mathbb{Z}_p^*}$

$\Rightarrow (\mathbb{Z}_p^*, *)$ possesses invertibility.

✓ (iii) Identity – Yes

$\exists I \in \mathbb{Z}_p^*$ s.t. $\forall x \in \mathbb{Z}_p^*$, $x * I = I * x$

$\Rightarrow (xI) \bmod p = (Ix) \bmod p$

Set $I = 1 \Rightarrow x \bmod p = x \bmod p \quad \forall x \in \mathbb{Z}_p^*$

$\Rightarrow (\mathbb{Z}_p^*, *)$ possesses the identity property.

✓ (iv) Closure – Yes: $(x \in \mathbb{Z}_p^*)$ and $(y \in \mathbb{Z}_p^*) \Rightarrow \underbrace{(xy) \bmod p \in \mathbb{Z}_p^*}_{\text{definition of mod}}$, so Yes.

$\Rightarrow (\mathbb{Z}_p^*, *)$ is a group. □

(III) $a, b, c, d, e \in \mathbb{Z}$ and $a, b, c > 0$

Claim: If $\gcd(a,b) = c$ & $\gcd(c,d) = e$, then e divides a.

Proof: We have $\gcd(a,b) = c$ & $\gcd(c,d) = e$,

and $a, b, c > 0$.

(e divides a) $\equiv (\exists x \in \mathbb{Z} : ex = a)$

$\equiv (a \bmod e \equiv 0)$, and we're looking for this x

We know

$(\gcd(a,b) = c) \Rightarrow ((a \bmod c \equiv 0)$ and $(b \bmod c \equiv 0))$

and

$(\gcd(c,d) = e) \Rightarrow ((c \bmod e \equiv 0)$ and $(d \bmod e \equiv 0))$

Since we have $c \bmod e \equiv 0$, $\exists\, y \in \mathbb{Z} : ey = c$

$\Rightarrow$ we also have some $z \in \mathbb{Z} : cz = a$

$\Rightarrow$ we can say $c = \frac{a}{z}$ since we know z is a factor of a.

$\Rightarrow \frac{a}{z} = ey$ for some $y, z \in \mathbb{Z}$

$\Rightarrow a = eyz \Rightarrow$ set $x = yz$ and we found an $x \in \mathbb{Z}$ such that $a = xe$

$\Rightarrow$ **e divides a** □

5 (W) - Disproved

$\exists\, n \in \mathbb{Z}$, $n \geq 1$, and $a_1, \dots, a_{n+1} \in \mathbb{Z}$

Such that for every $i, j \in \{1, \dots, n+1\}$, $i \neq j$, we have

$$(|a_i - a_j|) \bmod n > 0$$

$$\Rightarrow |(a_i - a_j) \bmod n| > 0$$

$$\Rightarrow |(a_i - a_j) \bmod n| \not\equiv 0$$

We have $\|\{1, \dots, n+1\}\| = n+1$, and

we have $\|\mathbb{Z}_n^*\| = n-1$.

We want to see if $\exists\, n \in \mathbb{Z}$, $n \geq 1$

with $a_1, \dots, a_{n+1} \in \mathbb{Z}$ such that

every possible $|a_i - a_j| \bmod n \in \mathbb{Z}_n^*$ $\forall\, i \neq j$, $i, j \in \{1, \dots, n+1\}$

Buth if $\|\{1, \dots, n+1\}\| = n+1$, then the size of

the # of all possible $|a_i - a_j| \bmod n$ we can get

is n. By Pigeonhole Principle, since

$n > n-1$, $\exists$ some difference $|a_i - a_j| \bmod n \notin \mathbb{Z}_n^*$,

and thus $\nexists\, n \in \mathbb{Z}$, and $n \geq 1$ and $n+1$ integers

such that for every $i, j \in \{1, \dots, n+1\}$, $i \neq j$, $|a_i - a_j|$ does not divides n.

(7) (6)

DetectPrime:

Input:= binary representation of $x \in \mathbb{Z}$

If $x=2$, output PRIME

For $i = 2$ to $\lceil \sqrt{x} \rceil$:

if i divides x, output NOT PRIME

output PRIME

Claim: DetectPrime does not run in polynomial time.

Explanation:

The input is a binary representation, its length = # of bits.

For some large $x=n$,

the for loop runs $(\sqrt{x}-1)$ times, i.e.

every iteration, we do integer division, trying to divide x,

(worst case here is a very large prime $x=n$, in which case loop runs the whole time)

Now, the rate

If we add one more bit to some input x, n becomes $n+1$, but we might be doing double the iterations in the for loop (if x is prime), i.e. 2^n operations, which is exponential, not polynomial (poly(n) vs 2^n). □

(7) We have:

$\phi(N) = 2^{n-1}$

$N = 2^n$

$ed \equiv 1 \bmod 2^{n-1}$

Euler's: $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a,n) = 1$

take an odd public key e,

$\gcd(e, 2^n) = 1$ indeed.

$\Rightarrow e^{2^{n-1}} \equiv 1 \bmod 2^n$

$\Rightarrow (e^{2^{n-1}})e^{-1} \equiv \underbrace{e^{-1} \bmod 2^n}$

d = modular inverse

$\Rightarrow e^{2^{n-1}-1} \equiv d \bmod 2^n$

$\Rightarrow e^{2^{n-1}-1} = x2^n + d$

$\Rightarrow \boxed{d = e^{2^{n-1}-1} - x2^n}$

Now we have a formula for

$d := e^{2^{n-1}-1} - x2^n$.

A:

Known: e, N Unknown: d

$y = e^{2^{n-1}-1}$, N/2

Loop: { x=0

$d = y - x2^n$ N

check $ed \equiv 1 \bmod 2^n$

if yes, output d

else, x++ & repeat loop.

}

A should efficiently terminate every time, since the amount of iterations done here is polynomial w respect to the inputs e, N

(I'm not sure if it terminates at a specific x, or)