Cryptography Notes

Rasmus Kirk Jakobsen Mikkel Skafsgaard Berg

January 21, 2024

Note that these notes are based on the 2023v3 version of the cryptography book.

Contents

Information theory and Cryptography (Chapter 5)

Disposition (Kirk):

- Perfect Security
- Entropy
- Unicity Distance
  - H_L
  - Redundancy
  - Spurious Keys
  - Unicity Distance

Perfect Security

Definition 5.1 (Perfect Security): $P[x|y] = P[x]$ △

Theorem - $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{P}|$: If you have perfect security then $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{P}|$. △

Entropy

Definition 5.6 (Entropy):

$$H(X) = \sum_{i=1}^{n} p_i \log_2(1/p_i)$$

TLDR: The entropy $H(X)$ can be described as:

- How many bits we need to send on average to communicate the value of X.
- The amount of uncertainty you have about X before you are told what the value is. △

Theorem 2.4 (Jensen's inequality):

$$\sum_{i=1}^{n} p_i f(x_i) \leq f(\sum_{i=1}^{n} p_i x_i)$$

- f must be concave
- Equality **iff** all x_i is equal △

Theorem 5.7 (Entropy Bounds):

$$0 \leq H(X) \leq \log_2(n)$$

.

- $H(X) = 0$ **iff** one value X has probability 1 (and the others 0).
- $H(X) = log_2(n)$ **iff** it is uniformly distributed, i.e., all probabilities are $1/n$. △

Proof: We need to prove the following:

- $H(X) > 0$
- $H(X) = 0$ **iff** a single $p_i = 1$ and all other $p_j = 0$
- $H(X) < \log_2(n)$
- $H(X) = \log_2(n)$ **iff** X is uniformly distributed. □

Conditional Entropy

Definition 5.9: Given the above definition of $H(X \mid Y = y_j)$, we define the conditional entropy of X given Y to be:

$$H(X \mid Y) = \sum_j P[Y = y_j] H(X \mid Y = y_j)$$

△

Entropy of Random Variables in Cryptography

Theorem 5.11: $H(K \mid C) = H(K) + H(P) - H(C)$ **iff** deterministic encryption function △

Unicity Distance

Definition - Average Bits of Information per Letter: $H_L = \lim_{n \mapsto \infty} H(P_n)/n$ △

Definition - Redundancy:

$$R_L = \frac{\log(|\mathcal{P}|) - H_L}{\log(|\mathcal{P}|)} = 1 - \frac{H_L}{\log(|\mathcal{P}|)}$$

△

Definition - Spurious Keys: A spurious key *seems* to be the correct key but is not. △

Definition - Number of Spurious Keys:

$$sp_n = \sum_{\boldsymbol{y} \in \mathcal{C}^n} P[y](|K(\boldsymbol{y})| - 1) = \sum_{\boldsymbol{y} \in \mathcal{C}^n} P[y]|K(\boldsymbol{y})| - 1$$

$$K(\boldsymbol{y}) = \{K \in \mathcal{K} \mid P[D_K(\boldsymbol{y} > 0)]\}$$

$$P[y] = \sum_{(x,K):E_K(x)=y} P[x]P[K]$$

△

Definition 5.12: The unicity distance n_0 of a cryptosystem is the minimal length of plaintexts such that $sp_{n_0} = 0$, if such a value exists, and ∞ otherwise.
△

Theorem 5.13: Assume we have a cryptosystem with deterministic encryption function, where the plaintext and ciphertext alphabets have the same size ($|\mathcal{C}| = |\mathcal{P}|$), and where keys are uniformly chosen from $\mathcal{K}$. Assume we use the system to encrypt sequences of letters from language L. Then

$$n_0 \geq \frac{\log(|\mathcal{K}|)}{R_L \log(|\mathcal{P}|)}$$

TLDR: If we reuse keys, our unconditional security will always be gone, once we encrypt enough plaintext under the same key. The only exception is the case where $R_L = 0$ which leads to n_0 being ∞. Which makes sense, if every sequence of characters is a plaintext that can occur, the adversary can never exclude a key. △

Proof: We start by unfolding the definition of $H(K \mid C_n)$ using Definition 5.9:

$$H(K \mid C_n) = \sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{C}_n}} P[C_n = \boldsymbol{y}] H(K \mid C_n = \boldsymbol{y})$$

First, note that given some ciphertext $\boldsymbol{y}$, the key K will have some conditional distribution, but of course only values in $K(\boldsymbol{y})$ can occur. Therefore $H(K|C_n = \boldsymbol{y}) \leq \log_2(|K(\boldsymbol{y})|)$:

$$\begin{aligned} H(K \mid C_n) &\leq \sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{C}_n}} P[C_n = \boldsymbol{y}] \log_2(|K(\boldsymbol{y})|) \\ &\leq \log_2 \left(\sum_{\boldsymbol{y} \in \boldsymbol{\mathcal{C}_n}} P[C_n = \boldsymbol{y}]|K(\boldsymbol{y})| \right) && \text{(Definition 2.4 - Jensen's Inequality)} \\ &\leq \log_2(sp_n + 1) && \text{(Definition - Number of Spurious Keys)} \end{aligned}$$

Now we want to simplify $H(K \mid C_n)$. We start by applying Theorem 5.11:

$$H(K \mid C_n) = H(K) + H(P_n) - H(C_n)$$

Observe that $H(C_n) \geq log(|C|^n) = n \log(|\mathcal{P}|)$. Moreover, recalling the definition on H_L, let us assume that we take n large enough so that $H(P_n) \approx nH_L$.

$$\begin{aligned} H(P_n) &\approx nH_L \\ &\approx n(\log(|\mathcal{P}|)(1 - R_L)) && \text{(Definition - Redundancy)} \end{aligned}$$

Now we try to find $H(K \mid C_n)$

$$\begin{aligned} H(K \mid C_n) &= H(K) + H(P_n) - H(C_n) \\ &\geq H(K) + H(P_n) - n \log(|\mathcal{P}|) && \text{(From our observation of } H(C_n)) \\ &\approx H(K) + n \log(|\mathcal{P}|)(1 - R_L) - n \log(|\mathcal{P}|) && \text{(From our estimate of } H(P_n)) \\ &= H(K) + n \log(|\mathcal{P}|) - n \log(|\mathcal{P}|)R_L - n \log(|\mathcal{P}|) \\ &= H(K) - n \log(|\mathcal{P}|)R_L \\ &= \log(|\mathcal{K}|) - n \log(|\mathcal{P}|)R_L && \text{(Theorem 5.7, } K \text{ is uniform)} \\ H(K \mid C_n) &\geq \log(|\mathcal{K}|) - n \log(|\mathcal{P}|)R_L \end{aligned}$$

Combining our equations, setting $sp_n = 0$ and solving for n:

$$\begin{aligned} \log(|\mathcal{K}|) - n \log(|\mathcal{P}|)R_L &\leq \log_2(sp_n + 1) \\ \log(|\mathcal{K}|) - n \log(|\mathcal{P}|)R_L &\leq \log_2(0 + 1) \\ n \log(|\mathcal{P}|)R_L &\leq \log(|\mathcal{K}|) \\ n &\leq \frac{\log(|\mathcal{K}|)}{\log(|\mathcal{P}|)R_L} \end{aligned}$$

So $n_0 \leq \frac{\log(|\mathcal{K}|)}{\log(|P|)R_L}$. □

Symmetric (secret-key) cryptography (Chapter 4.1 + 6)

Disposition (Kirk):

- Symmetric Cryptosystems
- CBC
- PRF
- CPA
- CPA security proof for CBC/CTR

Symmetric Cryptosystems

- $G :: \mathcal{K}$
- $E :: \mathcal{P} \to \mathcal{C}$
- $D :: \mathcal{C} \to \mathcal{P}$

$\forall x \in \mathcal{P} : x = D_K(E_K(x))$

PRF Security

We want the $Adv_A(O_R, O_I) \leq \epsilon$.

More formally, we want our PRF's to be secure as given by the following definition:
Definition - PRF Security: $\left\{ f_K \mid K \in \{0,1\}^k \right\}$ is (t', q', ϵ') PRF-secure if $Adv_A(O_R, O_I) \leq \epsilon$ $\triangle$

CPA Security

Definition - Chosen-Plaintext Attack(CPA)-security: (G, E, D) is (t, q, μ, ϵ) CPA-secure if $\mathrm{Adv}_A(O_R, O_I) \leq \epsilon$

μ denotes the number of bits an adversary encrypts!

$\triangle$

Theorem: If (G', E', D') is (t', q', ϵ') PRF-secure then (G, E, D) using CBC is $(t, q\mu, \epsilon)$ CPA-secure for any q, and for

$$\epsilon = \epsilon' + \left(\frac{\mu}{n}\right)^2 \cdot \frac{1}{2^n} = \epsilon' + \frac{\mu^2}{n^2 \cdot 2^n}$$

provided that

$$t \leq t', \quad \frac{\mu}{n} \leq q'$$

$\triangle$

Proof:
We start by introducing the *hybrid* oracle to the game

Right off the bat, since the *only* difference between the hybrid and real game is that E_K is replaced with R, we must have:

$$\begin{aligned} Adv_A(O_{real}, O_{hybrid}) &= |p(A, real) - p(A, hybrid)| \\ &\leq \epsilon \end{aligned}$$

If this was not the case, A could be used to distinguish between E_K and a random function with advantage greater than ϵ, contradicting our assumption that (G, E, D) was PRF-secure.

Now note that if we are in the ideal case, the oracle does *not* use CBC, but simply outputs $N + 1$ blocks, where N is the number of blocks in the input. It should now be difficult for A to distinguish between

the ideal and hybrid case, since the hybrid case outputs a concatenation of random blocks, also yielding $N+1$ random blocks, *UNLESS* a certain bad event happens. We define BAD as; if at any point during the hybrid game, the function R receives an input that it has received before in this game. In this case we will have an input collision, which will yield a repeated block. This could hint A that he is in the hybrid case. Therefore, his advantage in distinguishing hybrid from ideal must be bounded by:

$$|p(A, hybrid) - p(A, ideal)| \leq Pr(BAD)$$

If we add our two inequalities, we get:

$$\begin{aligned} |p(A, real) - p(A, ideal)| &\leq |p(A, real) - p(A, hybrid)| + |p(A, hybrid) - p(A, ideal)| \\ Adv_A(O_{real}, O_{ideal}) &\leq \epsilon + Pr(BAD) \end{aligned}$$

So now, we just have to estimate $Pr(BAD)$ by bounding it. Let M_j be the event that a collision occurs after j calls to R. Clearly $P(M1) = 0$. Using the Law of Total Probability, we have that:

$$\begin{aligned} P[M_j] &= P[M_j|M_{j-1}]P[M_{j-1}] + P[M_j|\neg M_{j-1}]P[\neg M_{j-1}] \quad \text{(Law of Total Probability)} \\ &\leq P[M_{j-1}] + P[M_j|\neg M_{j-1}] \\ &= P[M_{j-1}] + \frac{(j-1)}{2^n} \end{aligned}$$

The last probability on the right hand side is equal to $\frac{(j-1)}{2^n}$: First, since M_{j-1} did not occur we have seen $j-1$ different inputs before. Second, the new input nr. j is the XOR of some message block and an independently chosen random block (either a y0-value chosen by the oracle or an output from R), it is therefore uniformly chosen. We conclude that in fact

$$P[M_j] \leq (1 + 2 + \ldots + (j-1)) \leq \frac{j^2}{2^n}$$

Now we've provided a bound for all j's (calls to R). Since the total number of calls is at most μ/n, we can replace j with μ/n. Thus it follows that $P(BAD) \leq \frac{\mu^2}{n^2 \cdot 2^n}$ and we are done.

□

Public-key cryptography from Factoring (Chapter 7 & 8)

Disposition (Kirk):

- RSA
  - Specification
  - Decryption
  - PCRSA and CPA-security
  - CCA
  - OAEP

RSA:

Definition The RSA algorithm:

- G:
  1. On input (even) security parameter value k, choose random $k/2$-bit primes p, q, and set $n = pq$.
  2. Select a number $e \in Z^*_{(p-1)(q-1)}$ and set $d = e - 1 \bmod (p-1)(q-1)$.
  3. Output public key $pk = (n, e)$ and secret key $sk = (n, d)$. For RSA, we always have $\mathcal{P} = \mathcal{C} = Z_n$.
- E: $E_{(n,e)}(x) = x^e \bmod n$
- D: $D_{(n,d)}(y) = y^d \bmod n$

$\triangle$

Decryption: $D_{(n,d)}(E_{(n,e)}(x)) = x$

Proof: We use the Chinese Remainder Theorem to get $x = x^{ed} \bmod n \implies x = x^{ed} \bmod q \wedge x = x^{ed} \bmod p$

Then prove:

Case: $x = 0$

$$0 = x = x^{ed}$$

Case: $x \neq 0$ **modulo** p

$$\begin{aligned} x &= x^{ed} \\ &= x^{ed-1}x \\ &= x^{(p-1)a}x \\ &= (x^{(p-1)})^a x \\ &= 1^a x \\ &= x \bmod p \end{aligned}$$

Case: $x \neq 0$ **modulo** q

$$\begin{aligned} x &= x^{ed} \\ &= x^{ed-1}x \\ &= x^{(q-1)b}x \\ &= (x^{(q-1)})^b x \\ &= 1^b x \\ &= x \bmod q \end{aligned}$$

$\square$

Definition: The PCRSA algorithm

- G: G_{RSA}.
- E: $b \in \mathbb{B}$, $x_b \in_R \mathbb{Z}_n : \text{lsb}(x_b) = b$, $E^{(n,e)}_{RSA}(b) = x_b^e \bmod n$
- D: $D_{(n,d)}(y) = \text{lsb}(D^{(n,d)}_{RSA}(y)) \bmod n$

△

Theorem: PCRSA is *almost* CPA secure: If you can extract the least significant bit with certainty of x given y then you have a contradiction of the RSA assumption. △

Proof: Define the following two functions:

$$P(y) = lsb(y^d), \qquad H(y) = \begin{cases} 0, & \text{if } 0 \leq x \leq n/2 \\ 1, & \text{otherwise} \end{cases}$$

Note that:

$$P(y) = H(2^{-e}y), \qquad H(y) = P(2^e y)$$

Now we can perform binary search for x given P in $k = \lfloor \lg(n) \rfloor$ queries. This is done by constructing H and doubling y ($y' = 2^e y$).

□

CCA & OAEP

Definitions: CCA Security: If you can extract the least

Case: O_I:

1. A may submit an input string y to O_I, and O_I will return $D_{sk}(y)$ to A. This is repeated as many time as A wants.
2. A computes a plaintext $x \in \mathcal{P}$ and gives it to O_I. The oracle responds with $y_0 = E_{pk}(r)$, where r is randomly chosen in $\mathcal{P}$ of the same length as x.
3. A may now again submit an input string y to O_I, the only restriction is that y must be different from y_0. O_I will return $D_{sk}(y)$ to A. This is repeated as many time as A wants.

Case: O_R:

1. A may submit an input string y to O_R, and O_R will return $D_{sk}(y)$ to A. This is repeated as many time as A wants.
2. A computes a plaintext $x \in \mathcal{P}$ and gives it to O_R. The oracle responds with $y_0 = E_{pk}(x)$, where r is randomly chosen in $\mathcal{P}$ of the same length as x.
3. A may now again submit an input string y to O_R, the only restriction is that y must be different from y_0. O_R will return $D_{sk}(y)$ to A. This is repeated as many time as A wants.

△

Definitions: OAEP:

$E'_{pk} :: \mathbb{B}^k \to \mathbb{B}^a$
$k_0, k_1 : k_0 + k_1 < k$
$E_{pk} :: \mathbb{B}^n \to \mathbb{B}^b$, $n = k - k_0 - k_1$
$G :: \mathbb{B}^{k_0} \to \mathbb{B}^{n+k_1}$, $H :: \mathbb{B}^{n+k_1} \to \mathbb{B}^{k_0}$

Encryption E_{pk}:

1. Choose $r \in_R \mathbb{B}^{k_0}$.
2. Compute $s = G(r) \oplus (x + 0^{k_1}), t = H(s) \oplus r, w = s + t$
3. Let the ciphertext be $y = E'_{pk}(w)$.

Decryption D_{pk}:

1. s, t from $D'_{pk}(y) = w = s + t$.
2. $r = t \oplus H(s)$
3. x, s_0 from $G(r) \oplus s$
4. Check $s_0 \stackrel{?}{=} 0^{k_1}$
5. Output x $\triangle$

Public-key cryptography based on discrete log and LWE (Chapter 9 & 10, definition of CPS security in chapter 8)

Disposition (Kirk):

- DL, DH, DDH
- El Gamal
  - CPA
- Elliptic Curves

DL, DH, DDH

Definition: DL, DH, DDH:

- DL: Given α^a, find a
- DH: Given α^a, α^b, find α^{ab}
- DDH: Given $\alpha^a, \alpha^b, \alpha^c$, find $\alpha^{ab} \stackrel{?}{=} \alpha^c$

△

Theorem: Hardness of DL, DH, DDH: DL $\geq$ DH $\geq$ DDH △

Proof:

- Solving DL means solving DH: Given α^a, α^b, solve DL for a, then $(\alpha^b)^a = \alpha^a b$.
- Solving DH means solving DDH: Given $\alpha^a, \alpha^b, \alpha^c$ solve DH for α^{ab}, then $\alpha^{ab} \stackrel{?}{=} \alpha^c$.

□

El Gamal

Definition: Diffie-Hellman:

1. A sends B α^a
2. B sends A α^b
3. Both compute secret α^{ab}

Definition: El Gamal

Space:

- $\mathcal{P} = G$
- $\mathcal{C} = G \times G$

Algorithm:

- GGen(k): $G, \alpha \in G$
- G(k): $a \in_R \mathbb{Z}_t, pk = (GGen(k), \beta = \alpha^a), sk = a$
- E($x \in G$): $r \in_R \mathbb{Z}_t, y = (\alpha^r, \beta^r m)$
- D(c, d): x = $c^{-a}d$

Theorem: El Gamal Decryption Works △

Proof:

$$\begin{aligned} c^{-a}d &= (\alpha^r)^{-a}\beta^r m \\ &= \alpha^{-ar}\alpha^{ar}m \\ &= m \end{aligned}$$

□

Theorem: Under the DDH assumption El Gamal is CPA secure: △

Proof: We assume that there exists and adversary A that breaks CPA with advatage $> \epsilon$. Then we construct a subroutine B that uses A to break DDH using A, this will lead to a contradiction.

$$\begin{aligned} D(\alpha^b, \alpha^c m) &= \alpha^{-ab}\alpha^c m \\ &= \alpha^{c-ab} m \end{aligned}$$

Only if $c = ab$ do we get m. So we simply return the check $D(\alpha^b, \alpha^c m) \stackrel{?}{=} m$. This will have the same advantage as A so we have advantage $> \epsilon$.

□

Elliptic Curves

Definition: Elliptic Curves Define a function:

$$y^2 = x^3 + ax + b : 4a^3 + 27b^2 \neq 0$$

Addition of $P + Q = (x_1, y_1) + (x_2, y_2)$:

- $x_1 \neq x_2$: $P + Q = R$ (regular)
- $x_1 = x_2 \wedge y_1 = -y_2$: $P + Q = \mathcal{O}$ (above)
- $x_1 = x_2 \wedge y_1 = y_2$: $P + Q = 2P$ (tangent)

$$E_{a,b,p} = \{(x, y) \mid y^2 = x^3 + ax + b \bmod p\} \cup \mathcal{O}$$

△

Theorem 9.11 (Hasse): The order of N of $E_{a,b,p}$ satisfies $p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$ △

Definition: Elliptic Curves for El Gamal

- $E(m)$: $P \in_R E_{a,b,p}, (E_{EG}(P), H(P) \oplus m)$
- $D(c, d)$: $(H(D_{EG}(c)) \oplus d = H(P) \oplus (H(P) \oplus m)$

△

Symmetric authentication and hash functions (Chapter 11)

Disposition (Kirk):

- Definition of Collision intractible hash functions
  - Construction from DL
  - Implies one-way
- Merkle-Damgård
- CMA

Collision Intractible Functions

Definition: Collision Intractible Functions:

- $\mathcal{H}(k)$ produces $h(x) :: \mathbb{B}^* \to \mathbb{B}^k$
- Security:
  - Second Preimage Attack: $h(m_1) = h(m_2) : m_1 \neq m_2$
  - Collision Attack: For *any* m_1, m_2, $h(m_1) = h(m_2) : m_1 \neq m_2$ △

Definition: Hash Functions based on factoring and discrete log:

- $\mathcal{H}(k)$: $p = 2q + 1$ where q is a $k-1$-bit prime, α, β of order q in $\mathbb{Z}_p^*$
  - $h(m_1, m_2) = \alpha^{m_1}\beta^{m_2} \bmod p$
- Security. Assume collision:
  - $h(m_1, m_2) = h(m_1', m_2')$ where $(m_1 \neq m_1' \vee m_2 \neq m_2')$ then $\alpha^{m_1}\beta^{m_2} = \alpha^{m_1'}\beta^{m_2'}$
  - Then $\alpha = \beta^{(m_2 - m_2')(m_1 - m_1')^{-1} \bmod q} \bmod p$ △

Lemma 11.2: Collisions-intractable hash functions are one-way: Given function $h :: \mathbb{B}^{2k} \to \mathbb{B}^k$, and assume we are given an algorithm A running in time t that, when given $h(m)$ for uniform m, returns a preimage of $h(m)$ with probability ϵ. Then a collision for h can be found in time t plus one evaluation of h and with probability at least $\epsilon/2 - 2 - k - 1$.

Proof:

- $P[m \text{ is only child}] \leq 2^{-k}$
- $P[m \text{ is only child and } A\text{is succesful}] = P[G] \geq \epsilon - 2^{-k}$
- $P[A \text{ finds collision}] = P[C] = \epsilon$
- $P[C|G] \geq 1/2$

$$P[C] \geq P[C \cap G] = P[C|G]P[G] \geq 1/2 \cdot P[G] \geq (\epsilon - 2^{-k})/2$$

Merkle-Damgård Construction

Theorem 11.3 (Merkle-Damgård): If there exists a collision-intractable hash function generator $\mathcal{H}'$ producing functions with finite input length $m > k$, then there exists a collision- intractable generator $\mathcal{H}$ that produces functions taking arbitrary length inputs. △

Proof:
Case: $m - k > 1$**:**
$v = m - k - 1 > 0$
$\mathcal{H}'(k) = f :: \mathbb{B}^m \to \mathbb{B}^k$

1. Split x into v-bit blocks $x_1, x_2, ...x_n$ pad x_n with zeros if needed.
2. Add x_n containing the number of bytes used to pad x_n.
3. Define m-bit blocks $z_1, z_2, ..., z_{n+1}$

- $z_1 = 0^k + 1 + x_1$
- $z_i = f(z_{i-1}) + 0 + x_i$

4. Define $h(x) = f(z_{n+1})$

Case: $m - k = 1$**:**

1. Same arguments, but fails on last check **iff** x' is a suffix of x
2. $H(x) = H(E(x))$ where E is a suffix free encoding functions.
3. E(x) $= 0, 1 + D(x)$ where $D(x)$ repeats each bit twice. □

MACs

Definition: MACs

- G: K
- $A(m) = s$
- $V(s, m) = acc \vee rej$

Where $V(A(m), m) = acc$. △

Definition: HMAC: Two 512 bit constants:

- $ipad$ = 3636...36
- $opad$ = 5C5C...5C

$$HMAC_K(m) = SHA1(\ (K \oplus opad) + SHA1((K \oplus ipad) + m))$$

△

Signature schemes (Chapter 12)

Disposition (Kirk):

- Definition of Signature and CMA
- RSA Signatures
- Schnorr
  - Cannot cheat
  - Signature Scheme from interactive game

Signature Schemes

Definition: MACs

- G: K
- $A(m) = s$
- $V(s, m) = acc \vee rej$

Where $V(A(m), m) = acc$. △

RSA Signatures

Definition: Simple RSA Signatures

- G: G_{RSA}
- $A(m) = D_{RSA}(m) = m^d = s$
- $V(s, m) = E_{RSA}(s) \stackrel{?}{=} m = s^d \stackrel{?}{=} m$

Not CMA secure! △

Definition: CMA Secure RSA Signatures

- G: $G_{RSA}, h = \mathcal{H}(k)$
- $A(m) = D_{RSA}(h(m)) = h(m)^d = s$
- $V(s, m) = E_{RSA}(s) \stackrel{?}{=} h(m) = s^d \stackrel{?}{=} h(m)$

Secure if we model the Full Domain Hash as a random function and under the RSA assumption. △

Schnorr Signature Scheme

Definition: The Schnorr ZK Interactive Game
$p, q : q|p-1, \alpha \in \mathbb{Z}_p^* : |\alpha| = q, \alpha = \alpha_0^{p-1/q}, \alpha_0$ is a generator for $\mathbb{Z}_p^*$
$pk = (p, q, \alpha, \beta = \alpha^a), sk = a$

$$
\begin{aligned}
P \to V &: c = \alpha^r \\
V \to P &: e \in_R \mathbb{Z}_q^* \\
P \to V &: z = (r + ae) \\
V &: \alpha^z \stackrel{?}{=} c\beta^e
\end{aligned}
$$

If P is honest:

$$\begin{aligned}\alpha^z &= c\beta^e \\ &= \alpha^r(\alpha^a)^e \\ &= \alpha^{r+ae} \\ &= \alpha^z\end{aligned}$$

△

Theorem: If P can reliably "cheat", then he knows a: If P can guess more than 1 e reliably, then he can easily calculate a. △

Proof: $e \neq e'$
$z = r + ae, \quad z' = r + ae'$
$\alpha^z = c\beta^e, \quad \alpha^{z'} = c\beta^{e'}$

$$\begin{aligned}\alpha^{(z-z')} &= \beta^{(e-e')} \\ \alpha^{(z-z')(e-e')^{-1}} &= \beta \\ \alpha^a &= \beta\end{aligned}$$

□

Definition: Fiat-Shamir on the Schnorr Interactive Game: We can use the Fiat-Shamir heuristic to go from interactive to non-interactive:

$$\begin{aligned}P \to V &: (e = h(c, m), z, c) \\ V &: \alpha^z \stackrel{?}{=} c\beta^e \;\&\&\; c \stackrel{?}{=} h(c, m)\end{aligned}$$

Alternatively:

$$\begin{aligned}P \to V &: (e, z) \\ V &: c = \alpha^z\beta^{-e} \\ V &: \alpha^z \stackrel{?}{=} c\beta^e \&\& \stackrel{?}{=} h(c, m)\end{aligned}$$

△

Definition: Fiat-Shamir on the Schnorr Interactive Game: We can use the Fiat-Shamir heuristic to go from interactive to non-interactive:

$$\begin{aligned}P \to V &: (e = h(c, m), z, c) \\ V &: \alpha^z \stackrel{?}{=} c\beta^e \;\&\&\; c \stackrel{?}{=} h(c, m)\end{aligned}$$

Alternatively:

$$\begin{aligned}P \to V &: (e, z) \\ V &: c = \alpha^z\beta^{-e} \\ V &: \alpha^z \stackrel{?}{=} c\beta^e \;\&\&\; e \stackrel{?}{=} h(c, m)\end{aligned}$$

△

Definition: Schnorr Signature Scheme:

- G: Output $pk = (h, p, q, \alpha, \beta = \alpha^a mod p)$ and $sk = a$.
- $A(m) = (e, z)$
- $V(s, m) = \alpha^z \stackrel{?}{=} c\beta^e \;\&\&\; e \stackrel{?}{=} h(c, m)$

Appendix

CPA

$$\begin{aligned}\epsilon &= \epsilon' + \left(\frac{\mu}{n}\right)^2 \cdot \frac{1}{2^n} \\ &= \epsilon' + \frac{\mu^2}{n \cdot 2^n}\end{aligned}$$

Solving for $1 = |\epsilon - \epsilon'|$:

$$\begin{aligned}1 &= |\epsilon - \epsilon'| \\ 1 &= \frac{\mu^2}{n \cdot 2^n} \\ n \cdot 2^n &= \mu^2 \\ \sqrt{n \cdot 2^n} &= \mu \\ \sqrt{n} \cdot 2^{n/2} &= \mu\end{aligned}$$

So if we encrypt much less than $2^{n/2}$ we are safe. We disard $\sqrt{n}$ since it is insignificant compared to $2^{n/2}$.

□

CPA

$$\begin{aligned}P[M_j] &= P[M_j|M_{j-1}]P[M_{j-1}] + P[M_j|\neg M_{j-1}]P[\neg M_{j-1}] \quad \text{(Law of Total Probability)} \\ &= \frac{P[M_j, M_{j-1}]}{P[M_{j-1}]}P[M_{j-1}] + P[M_j|\neg M_{j-1}]P[\neg M_{j-1}] \quad \text{(Bayes rule)} \\ &= P[M_j, M_{j-1}] + P[M_j|\neg M_{j-1}]P[\neg M_{j-1}] \\ &= P[M_j]P[M_{j-1}] + P[M_j|\neg M_{j-1}]P[\neg M_{j-1}] \\ &\leq P[M_{j-1}] + P[M_j|\neg M_{j-1}]\end{aligned}$$