CS 255 Cheat Sheet

Arjun Pandey

March 2024

1 Introduction

1. **Stream Ciphers**

   - $E(k,m) = m \in [G(k)[0,\ldots,|m|-1]]; D(k,c) = c \oplus [G(k)[0,|c|-1]]$

   - Security: PRG secure $\Rightarrow$ stream cipher "secure"

   - Attack on OTP and Stream Ciphers $\Rightarrow$ Two-time Attack.

   - Shannon Secrecy: cipher (E,D) has perfect secrecy if single CT reveals no "info" about PT s.t. $k \xleftarrow{r} K$.

   - Perfect secrecy implies that $|K| \geq |M|$.

2. **Pseudorandom Generators** (PRG).

   - Def: $G : X \to Y$ is a secure PRG if for every "eff" A: $\text{Adv}[A,G] \leq$ neg. $\left[= 2^{-80}\right]$

   - More generally over a finite set Ω, choose p_1, p_2 as distributions and we must see that:

$$\text{Adv}\left[(A,p_1,p_2)\right] := \left|\Pr\left[x \xleftarrow{r} p_1 : A(x) = 1\right] - \Pr\left[x \xleftarrow{r} P_2 : A(x) = 1\right]\right| \leq \text{ neg.}$$

   -

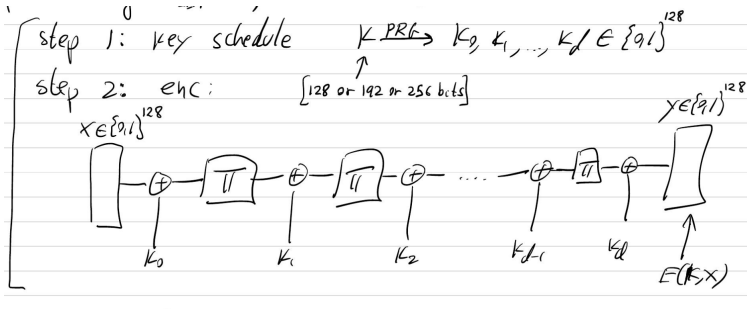
   - An important property of PRG is that they must be unpredicatable.

3. **Proof for Perfect Secrecy:**

   - Fix $c \in C$, then show that for every $m \in M$ there must exist at least one $k \in K$ s.t. $E(k,m) = c$

2 Block Ciphers

1a. **Abstract AES - Iterated Evan Mansour**



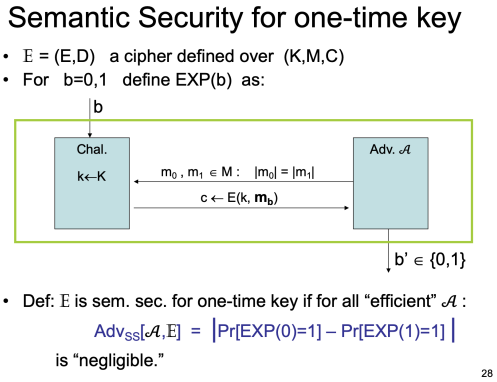
- Alg $D_{\text{IEM}}(k, y)$; apply π^{-1}& keys in reverse order.
- π is an invertible permutation function

1b. Security Theorem: If π is a fixed invertible function chosen at random and $k_0, \ldots, k_d \xleftarrow{r} \{0,1\}^n$, then $(E_{\text{IEM}}, D_{\text{IEM}})$ is a secure block cipher given $n \geq 256$ and $d \geq 10$.

2a. PRF: $F : K \times X \to Y$ is indistinguishable from random function in Funs $[x, y]$

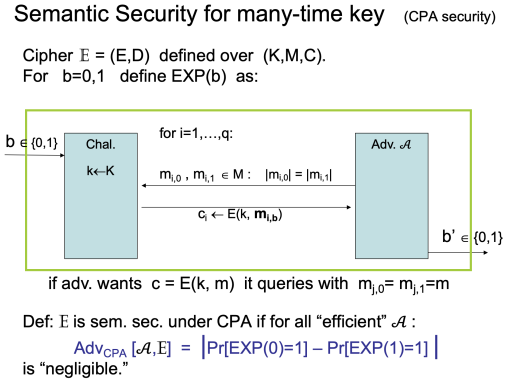
2b. PRP: $E : K \times X \to X$ is indistinguishable from random function and is efficiently invertible. A PRP over (K, X) is a PRF over (K, X, X)

3a. **Semantic Security for One-time Key**



- Goal is to guess whether the encryption was of m_1 or m_0.

3b. **CPA-Security** (for many-time keys)

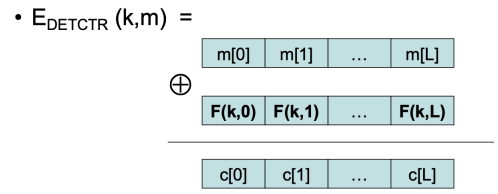


- Adversary sends multiple message pair queries i.e. $(m_{i,0}, m_{i,1})$.
- Goal is to guess what message is encrypted i.e. $0, 1$ for any i.
- Suppose, we want that $\text{Adv}_{\text{CPA}} \leq 1/2^{32}$. Then we need that $q^2 L/|X| \leq 1/2^{32}$.
- In AES: after 2^{32} CTs of length 2^{32}. Must change key.

4. Insecurity of RAW CBC without the last step with a different key. Look at the following adversary:
   (i) chose arbitrary $m \in X$
   (2) request tag for m. let $t = \text{RawCBC}(k, m) = F(k, m)$

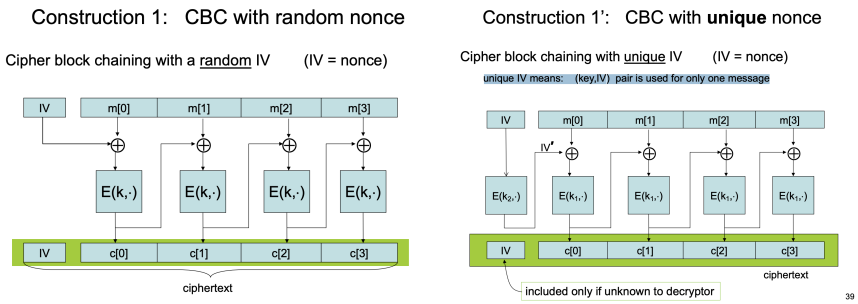
(3) output (msg, tag) forgery where msg= $(m, t \oplus m) \in X^2$, tag. $= t$
Then: RawCBC$(k, (m, t \oplus m)) = F(k, F(k, m) \oplus (t \oplus m)) = F(k, m) = t$

5. **Modes of Operation** (One-time Key)
   a. Stream Ciphers built from PRF (eg. AES). One-time pads.
   b. Deterministic Counter Mode from a PRF F.

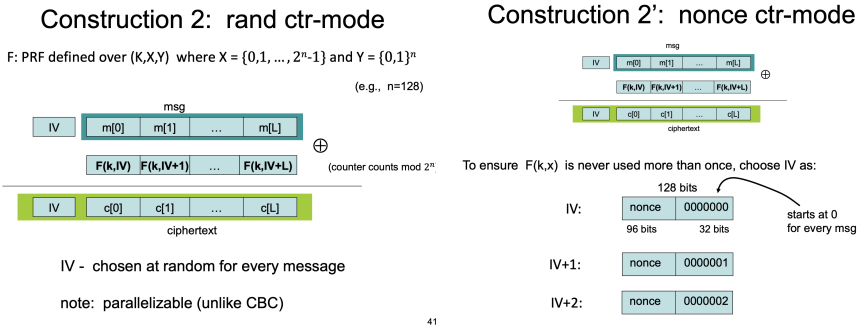


- Another example is stream ciphers built from PRF eg. AES.

6. **Modes of Operation** (Many-time Key)
   a. Nonce-based encryption (CBC):



   b. Nonce-based encryption (CTR mode):



3 MACs

1. Def: A MAC def over (K, M, T)
   is a pair of "eff" alg. (S, v) where
   - $S(k, m) \to t \in T$
   - $V(k, m, t) \to$ yes/ no
   st. $\forall k \in K, m \in M: \quad V(k, m, S(k, m)) =$ yes

2. Attacker's goal to break a MAC: existential forgery: produce valid (m,t) where $(m,t) \notin \{(m_1,t_1),\ldots,(m_9,t_9)\}$

3. Every secure PRF with a sufficiently large range gives a secure MAC:

   Let F be a PRF over (K,X,Y)

   Def: $I_F = (s,v)$ as: $\quad s(k,m) := F(k,m)$

   $$v(k,m,t) := \begin{cases} yes & \text{if } t = F(k,m) \\ no & \text{otherwise} \end{cases}$$

4. I_F is only a secure MAC when $F : (K,X,Y)$ has that $\frac{1}{|Y|}$ is negligible.

5. Suppose MAC I_F is built from a PRF and outputs n-bit tags $(y = [0,1]^n)$

   It's ok to truncate MAC to $w < n$ bits as long as $1/2^w$ is considered negligible

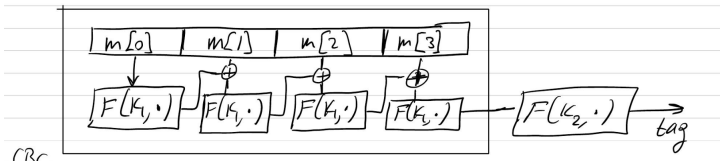
   (truncating a secure PRF is a secure PRF)

6. CBC-MAC, same as CBC cipher:

   In particular, for every q-query adv. A attacking F_{CBC} there is an adv. B (where time $(B) \approx$ time (A)) st.

   $$\text{PRF Adv}\,[A, F_{CBC}] \leq \text{PRF adv}\,[B,F] + \frac{q^2 \cdot L^2}{|X|}$$

   CBC-MAC is secure as long as $q \ll \sqrt{|X|}$



7. When MACs are not a multiple of block length: use one-to-one padding function i.e. pad with "$100\ldots0$" or add a dummy block of "$100\ldots0$".

   Insecure if you just pad with $0s$. Attack: ask for tag, t, for m get t for $m||0$.

4 Collision Resistant Hashing

1. Collision resistant hash function such that it must be hard to find m_0, m_1 where $m_0 \neq m_1$ s.t. $H(m_0) = H(m_1)$.

2. Definition: A function $H : M \to T$ is collision resistant if For all (explicit) "eff" algs. A :

   $CR\,\text{adv}[A,H] = \Pr[A \text{ outputs collision for } H]$ is negligible.

3a. Defining small MAC $\to$ Big-MAC:

   Let (S,v) be a secure MAC over (K,M,T) for short msgs, H: $M^{big} \to M$ be a CRH

   Define: (S',V') a MAC over $\left(K, M^{big}, T\right)$ where:

$$S'(k,m) := S(k,H(m))$$
$$V'(k,m,t) := V(k,H(m),t)$$

3b. **Theorem:** If (S, V) a secure MAC, H a collision resistant hashing function.

Then (S', V') is a secure MAC.

"Proof:" Suppose an adv. A attacks (S', V') :

(1) requests tag on $m_1, m_2, \ldots \in M$ and gets $t_1, t_2 \ldots$.

(2) outputs a Forgery $(m, t) \neq (m_i t_i), i = 1, 2, \ldots$

Then either:

(i) $\exists i : H(m) = H(m_i)$ and $m \neq m_i \Rightarrow$ Attack on CRHF.

or

(2) $\forall i = (H(m), t) \neq (H(m_i), t_i) \Longrightarrow$ Attack on (S, v)

4. **The Birthday Paradox**: Suppose you have n independet uniform RVs taking on B values. Theorem: when $n \geqslant 1.2\sqrt{B}$ then $\Pr[\exists i \neq j : r_i = r_j] \geq \frac{1}{2}$. Uniform RVs are woest case scenario.

5. Birthday attack: $\left[\begin{array}{l} \text{- choose random } m_0, m_1, \ldots, m_2 e/2 \in M \\ \text{- compute } H(m_0), \ldots, H(m_2 e/2) \\ \text{- Look for collision with } O(2^{\frac{e}{2}}) \end{array}\right.$

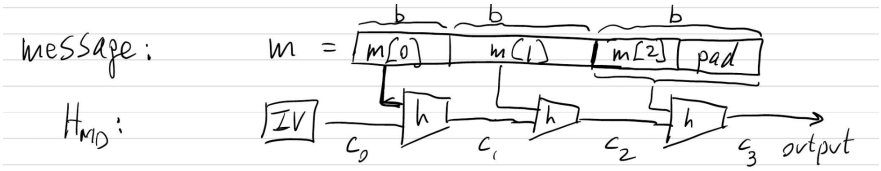
6. Typical hash value: $256 - \text{bit} \Rightarrow 2^{128}$ security.

7. Constructing a CRH using the Merkle-Damgard Construction: how to use CRH for small messages to construct a CRH for big messages.

Terminology:
(1) $h : \{0,1\}^b \times T \to T$ compression Function

(2) c_0, c_1, c_2, c_3 : chaining variables

(3) IV: Fixed initial value



8. Building HMAC from hash functions:

$$F_{\text{MAC}}(k, m) := H((k \oplus \text{ opad}) \| H(k \oplus i \text{ ipad } \| m))$$

9. Constructing Compression Function: Davies-Meyer

Let $E(K, X)$ be a Block cipher over (K, X)

$$h(m, c) = E(m, c) \oplus c$$

"Theorem." if E is "ideal" (random) then finding a collision takes time $\geq 2^{n/2}$

10. Collision resistance for Merkle-Damgard: if h is a CRH then H_{MD} is also C.R.H. so it suffices to construct a collision resistant compression function.

$$\begin{aligned} &M: \quad IV = c_0, c_1, c_2, \ldots, c_t \\ &M': \quad \text{IV} = c'_0, c'_1, c'_2, \ldots, c'_r \\ &H_{MD}(M) = H_{MD}(M') \Rightarrow c_t = c'_r \Rightarrow \\ &h(M[t-1], c_{t-1}) = c_t = c'_r = h(M[r-1], c_{r-1}). \end{aligned}$$

Essentially, we see that if Merkle Damgard is not CRH then h is not CRH since we can look at the last block of h and prove collision. Proof by contrapositive.

11. Applications of CRH.

    1. Small MAC can be used to construct Big MAC

    2. Software Package Integrity using read only space to store hashes. Attacker can not find $F' \to H(F_s) = H(F')$ since it's CRH.

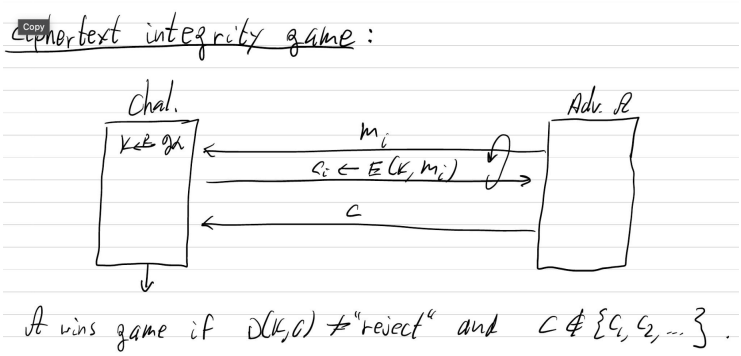
5 Authenticated Encryption

1. Important to combine integrity with confidentiality: CPA w/o integerity provides no confidentiality.

   - Def (E, D) provides auth. enc. if (E, D) is both CPA-secure and has Ciphertext Integrity.

1a. **Ciphertext Integrity**

   - Ciphertext Integrity Game:



   - Adversary can request ciphertexts on several messages $(c_0, \ldots, c_n)$. Goal to construct $c \notin (c_0, \ldots, c_n)$.

2. Options for AE: options:

   1. MAC-then-enc(TLS 1.0) $\quad t \leftarrow S(k_m, m), \quad c \leftarrow E(k_e, m \| l)$, send c
   2. enc-then-MAC(GCM): $c \leftarrow E(K_e, m), t \leftarrow S(K_m, C)$, send (c, t)
   3. enc-and-MAC(SSH): $t \leftarrow S(K_m, m), \quad c \leftarrow E(K_p, m)$, send (c, t)

3. Method 3 is insecure since MAC could info. about the message making it CPA insecure. Method 1. Can be insecure as well as seen in SSL 3.0.

4. Method 2 is called GCM uses nonce-based-ctr-mode and then MAC.

   Remember that nonce is used across cryptography to prevent replay attacks.

5. Authenticated Encryption w/ Associated Data. (add)

$$c \in E(k_e, m), \quad t \in s(k_m, \text{add} \| c), \quad \text{send}(c, t)$$

   Used in TLS to ensure that stuff that is in the clear like header doesn't get changed so example of additional associated data: header.

6. TLS:

   a. We have browser and server. Each side maintains unidirectional keys $k_{b \to s}, k_{s \to b}$.

b. Encryption is stateful and each side maintains two 64-bit counter.: WSC_B, WSC_S initialized to 0 at setup.

c. The counter of sent items is sent every time and is used as nonce.

d. WSC prevents replay, re-ordering, deletion.

6 Number Theory

1. $\mathbb{Z}_p$ is a set of $\{0, 1, 2, \ldots, p-1\}$.
2. Euclid's algorithm help computes the gcd of any two numbers i.e. find a, b such that: $an + bm =$ gcd(n, m).
3. $\gcd(h, m) = 1 \Rightarrow h, m$ are relatively prime.
4. x in $\mathbb{Z}_n$ has an inverse $\Leftrightarrow \gcd(x, n) = 1$
5. $\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\} = \{1, 2, \ldots, p-1\}$

6a. $\left[\begin{array}{l} \text{repeat:} \\ p = 2^{2047} \ldots, 2^{2048} - 1 \\ \text{until } 2^{p-1} = 1 \text{ in } \mathbb{Z}_p \\ \text{output } p \end{array}\right]$

6b. Fact: $\forall g \in G : g^{\text{order}(g)} = 1$.

- Corollary (Fermat): $\forall g \in G : \quad g^{|G|} = \left[g^{\text{order (g)}}\right]^{\frac{|G|}{\text{order(g)}}} = 1$

7. **Theorem (Lagrange):** $\forall g \in G :$ order(g) divides $|G|$.
8. Problem: given $h \in G$ and $1 < e < q$ find $y = h^{1/e}$ (ie. $y^e = h$)

   alg: $\left[\begin{array}{l} (1) \text{ compute } \alpha := \left(e^{-1} \bmod q\right) \in \mathbb{Z}_q \\ (2) \text{ output } y := h^\alpha \in G \end{array}\right]$

9. Time taken to factorize p in $\mathbb{Z}_p^* = e^{\sqrt[3]{ln(p)}}$ whereas in Elliptic Curves it's $\sqrt{p}$.

   So minimum $p \geq 2048$ bits in $\mathbb{Z}_p^*$. In EC it's 256 bits.

10. **Discrete Log Assumption:** $\forall$ "eff" adv. A $\Pr\left[A\left(g, g^\alpha\right) = \alpha\right]$ is "neg" when $\alpha \xleftarrow{r} \mathbb{Z}_q$.

10a. **Computational Diffie Hellman Assumption**: $\forall$ "eff" adv. A $\Pr\left[A\left(g, g^\alpha, g^\beta\right) = g^{\alpha\beta}\right]$ is "neg" when $\alpha, \beta \xleftarrow{r} \mathbb{Z}_q$.

11. Best known GNFS algorithm works in time $e^{\sqrt[3]{ln(p)}}$ so $p \geq 2048$ bits.

    In Elliptic curve cryptography, the best known algorithm runs in $\sqrt{p}$ and thus $p \geq 256$ bits.

12. Elliptic Curve Groups: $y^2 = x^3 + ax + b$

    - Given $p : (x_1, y_1)$ and $q = (x_2, y_2)$ there are efficient ways to compute $p + q$ on elliptic curve.

    - Consider set: $E_{a,b} = \{(x, y) \mid y^2 = x^3 + ax + b \in \mathbb{Z}_p\}$. If this set has prime order, then it must be cyclic and thus we can use it for Diffie Hellman.

13. **Euler's Theorem** Thm: (Euler) let $\varphi(n) = |\mathbb{Z}_n^*|$ then: $\forall x \in \mathbb{Z}_n^* : \quad x^{(|n)} = 1$ in $\mathbb{Z}_n$

14. Computing roots in G where G is a finite cyclic group of prime order q.
    - Given $h \in G$, for $1 < e < q$, find $y = h^{1/e}$.
    - compute $\alpha = e^{-1} \bmod q$.
    - output $y = h^{\alpha} \in G$.
15. Easy case of finding roots. When $gcd(e, p-1) = 1$, then c^e always exists in Z_p. For $d = e^{-1} \in Z_p^*$ then $d \cdot e = k(p-1) + 1$

7 Basic Key Exchange

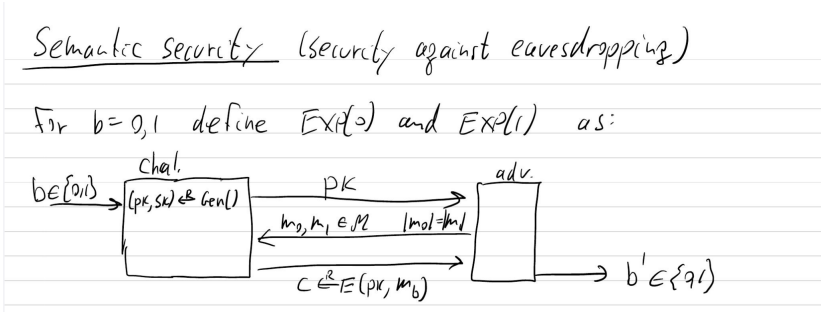
1. First Idea: **Trusted Third Party (TTP)**. Problems:
   (1) TTP is needed for every key exchange.
   (2) Knows all secret keys (backdoor heaven)
   (3) Insecure against active attacks like Replay Attacks:
   - attacker: record session between Alice & bank. eg.: $A \to B$: pay claire \$100.
   - attacker replays session to bank: Bank thinks Alice wants to pay claire again.
2. Key exhange without online TTP: Basic Diffie Hellman protocol:
   - Fix a large prime p (eg. 600 digits)
   - Fix an integer g s.t. $1 < g < p$.
   - $a \in^R \{1, \ldots, p-1\}, A \leftarrow g^a (\text{mod } p)$
   - $b \in^R \{1, \ldots, p-1\}, B \leftarrow g^b (\text{mod } p)$
   - Diffie Hellman Secret: $B^a = \left(g^b\right)^a (\text{mod} p) = k_{ab} = g^{ab} (\text{mod} p) \quad = A^b = (g^a)^b (\text{mod} p)$
   - CDH Assumption - in G: $\forall$ "eff" $A : \Pr\left[A\left(g, g^{\alpha}, g^{\beta}\right) = g^{\alpha\beta}\right) <$ is negligible i.e. the probablity of an adversary obtaining $g^{\alpha\beta}$ having g, g^{α}, g^{β} is ngeligible.
3. **Repeated Squaring** for computing $g^x (\text{mod } p)$
   Repeated squaring: $g^{13} = g^{(1101)_2} = g^8 \cdot g^4 \cdot g^1 (\text{mod } p)$: $O(\log(n))$ multiplications.
   In general: Let $x = x_n x_{n-1} \cdots x_1 x_0 \in \{0,1\}^{n+1}$ be binary rep. of x.
   alg.: $z \leftarrow 1, \quad y \leftarrow g$
   For $i = 0, 1, \ldots, n$:
   if $x_i = 1$ set $z \leftarrow z \cdot y \pmod p$
   $y \leftarrow y^2 \pmod p \qquad // \left(g, g^2, g^4, g^8, g^{16}, \ldots\right)$,
   output z

8 Public Key Cryptography

1. Semantic Security against eavesdropping: Def: $\epsilon = (G, E, D)$ is sem. secure if $\forall$ "eff" $\mathcal{A}$

$$Adv_{ss}[A, \varepsilon] := |\Pr[\text{Exp}(0) = 1] - \Pr[\text{EXP}(1) = 1]| < \text{"neg"}$$

   If E is deterministic, ε cannot be semantically secure.
   Semantic security $\Rightarrow$ A cannot distinguish $(pk, E(pk, k))$ from $(pk, E(pk, 0))$.



2. Application: **File Sharing** on the cloud. Here is how this can be done. Suppose Alice wants to upload F_1 to cloud to share with Bob:
   - Alice chooses $k_a \xleftarrow{r} \mathbb{K}$.
   - Alice choose another $k_1 \xleftarrow{r} \mathbb{K}$ for her file.
   - Alice uploads $E(k_a, k_1), E(k_1, F_1)$.
   - Suppose Bob wants to obtain this file now. He sends pk_b to Alice.
   - Alice uploads $E(pk_b, k_1)$ to the cloud. Bob recovers $D(sk_b, k_1)$ and then proceeds to $D(k_1, F_1)$

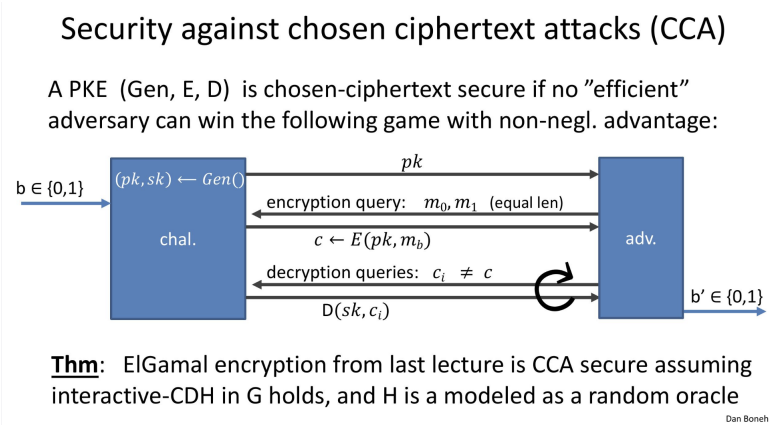
3. Constructing PKE through **ElGamal Encryption**:

   - G: Finite cyclic group of order q with generator $g \in \mathbb{G}$. (E_s, D_s) : sym. cipher over (K, M, C). H: $G^2 \to K$ a hash function.
   - Gen (): $\alpha \leftarrow \mathbb{Z}_q, \quad h := g^\alpha, \quad sk := \alpha, \quad pk := h \in G$
   - $E(pk, m)$: $\quad \beta \leftarrow \mathbb{Z}_q, \quad u := g^\beta, \quad v := h^\beta \quad (= g^{\alpha\beta})$
     $k := H(u, v) \in \mathcal{K} \longleftarrow$ key derived from DH secret
     $c := E_s(k, m)$
     output (μ, c)
   - $D(sk, (u, c))) : v = u^\alpha (= g^{\alpha\beta})$
     $k := H(u, v) \in \mathcal{K}$
     $m := D_s(k, c)$

4a. Security Theorems for El-Gamal:
   - Thm 1: (Gen, E,D) is semantically secure against eavesdroppping given:
     - (1) CDH holds in (G, g),
     - (2) (E_S, D_S) is sem. secure, and
     - (3) H is a secure key derivation function
   - Thm 2: (Gen, E,D) is chosen ciphertext secure.
     - (1) Interactive DH holds in (G, g)
     - (2) (E_s, D_s) provides Authenticated Encryption
     - (3) H is a "random oracle".

4b. **Chosen Ciphertext Attacks (CCA Security)**:
   - A, the adversary, has pk
   - A sends over messages m_0, m_1 of equal length and receives $c \leftarrow E(pk, m_b)$.
   - Adversary can also send $c_i \neq c$ to get back $D(sk, c_i)$

- It must guess $c = c_0$ or c_1 i.e. if it comes from the first message or second message with non-negligible probability



5. **Trapdoor Functions (TDFs)**

   - Definition: A trapdoor func. $X \to Y$ is a triple of efficient algs. (Gen, F, F^{-1})

   - Gen(): randomized alg. outputs a key pair (pk, sk)

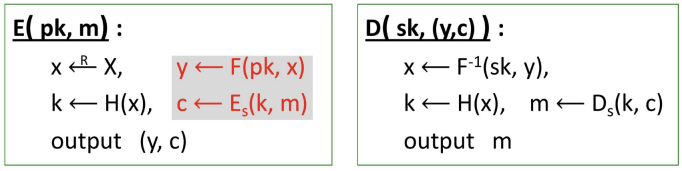
   - $F(pk, \cdot)$: deterministic algorithm that defines a function $X \to Y$

   - $\mathrm{F}^{-1}(\mathrm{sk}, \cdot)$: defines a function $\mathrm{Y} \to \mathrm{X}$ that inverts $\mathrm{F}(\mathrm{pk}, \cdot)$

   - (Gen, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function if it can be evaluated, but cannot be inverted without sk i.e.:

$$\mathrm{Adv}_{\mathrm{OW}}[\mathrm{A}, \mathrm{F}] = \Pr[\mathrm{x} = \mathrm{x}'] < \text{ negligible}$$

6. **Constructing PKE from TDFs**:



   - Security Theorem: If (Gen, F, F^{-1}) is a secure TDF, (E_s, D_s) provides auth. enc. and $\mathrm{H} : \mathrm{X} \to \mathrm{K}$ is a "random oracle" then $(\mathbf{Gen}, \mathbf{E}, \mathbf{D})$ is $\mathrm{CCA}^{\mathrm{ro}}$ secure.

   - Never encrypt by applying F and F^{-1} directly since F is deterministic and thus attackable.

7. **RSA Trapdoor Permuation**

   - Gen(): choose random distinct primes $p, q \approx 1024$ bits.

   - Set $\mathbf{N} = \mathbf{pq}$. Choose integers e,d s.t. e.d $=1 \pmod{\varphi(\mathbf{N})}$

   - Output $\mathrm{pk} = (\mathrm{N}, \mathrm{e}), \quad \mathrm{sk} = (\mathrm{N}, \mathrm{d})$

   - $\mathrm{F}(\mathrm{x}) = \mathrm{RSA}(\mathrm{x}) = \mathbf{x}^{\mathrm{e}} \quad$ in $\mathbb{Z}_n$

   - $\mathbf{F}^{-1}(\mathbf{sk}, \mathbf{y}) = y^d; \quad y^d = \mathbf{RSA}(\mathbf{x})^d = x^{ed} = x^{k\varphi(N)+1} = \left(x^{\varphi(N)}\right)^k \cdot x = x$

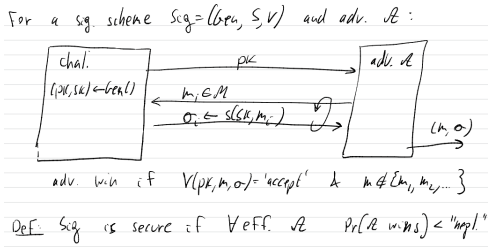
8. Simple Attack on Textbook RSA:
    1. Suppose k is 64 bits: $k \in \{0, \ldots, 2^{64}\}$. Eve (Adversary) sees: $c = k^e$ in $\mathbb{Z}_N$
    2. If $\mathbf{k = k_1 \cdot k_2}$ where $\mathbf{k}_1, \mathbf{k}_2 < 2^{34}$ (prob. $\approx 20\%$) then $\mathbf{c / {k_1}^e = {k_2}^e}$ in $\mathbb{Z}_N$
    3. Step 1: build table: $c/1^e, c/2^e, c/3^e, \ldots, c/2^{34e}$. time: 2^{34}.
    4. Step 2: for $k_2 = 0, \ldots, 2^{34}$ test if ${k_2}^e$ is in table. time: 2^{34}. Output matching (k1, k2)

9 Digital Signatures

1a. Binds document to author. Gives it "identity".

1b. Triplets of algorithms: (Gen, S, V) where we want that $V(pk, S(sk, m)) =$ yes and that $Gen() = pk, sk$.

2. Signature scheme is efficient if the adversary can't win the game of creating an **existential forgery**.
    - Attacker's Power: Chosen Message Attack.



3. Applications: Software Updates, Payments, Certificates.

3b. Extending Domains of Signature Schemes:
    - Let Sig $= (\text{Gen}, S, V)$ be a sig. scheme for short msg $m = \{0,1\}^{256}$.
    - Let $H : M^{\text{big}} \to M$ be a CRH (eg. SHA256)
    - $S^{\text{big}}(sk, m) : \sigma = S(sk, H(m))$
    - $V^{\text{big}}(pk, m, \sigma) = V(pk, H(m), \sigma)$
    - Security Theorem: If Sig is existentially unforgeable and H is a CRH, then $(S^{\text{big}}, V^{\text{big}})$ is secure.

4. Signature schemes using generic OWF eg: Lampart-Merkle signature. However, not commonly used due to large size of digital schemes.

5. We can build signature schemes using DLOG eg: ECDSA, BLS

6. We can build signature schemes using Trapdoor Permutations: RSA shown below.
    - TDFs are not sufficient here.

7. Application: **Payments System**: - PoS terminal sends Tx details and a nonce to Credit Card. - Credit Card sends back signature and certificate. Tx Details + nonce + sig + cert is sent to Visa servers for verification.

8. **RSA-FDH** (full-domain hash)
    - Gen: choose random primes p, q and get $e, d = 1 \bmod \varphi(n)$. Fix FDH: $M \to \mathbb{Z}_n$
    - output $pk = (n, e, FDH)$, sk $= (n, d, FDH)$
    - S(sk, $m \in M$) $= \sigma = [H(m)]^d$ in $\mathbb{Z}_n$

- V(pk, m, σ) = accept iff $\sigma^e = H(m)$ in $\mathbb{Z}_n$.
- Why hash messages in RSA-FDH? (Insecure if not)
  - **Attack 1 on FDH sigs**
    1. Choose $\sigma \in X$, set $m = F(pk, \sigma) = (\sigma^e \bmod n)$
    2. output (m, σ) (forgery).

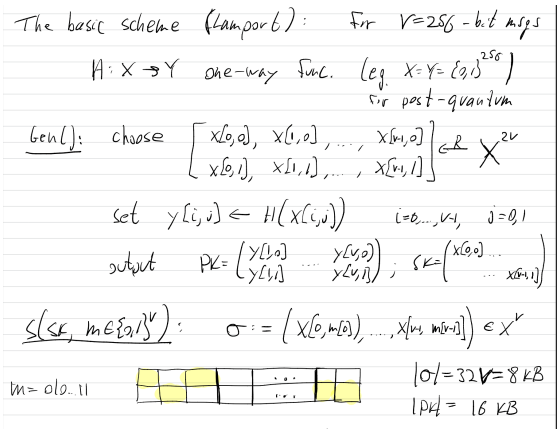
    This works because $m = \sigma^e$ and thus $F(\sigma, d) = \sigma$.
  - **Blinding Attack on RSA**
    1. choose $r \xleftarrow{r} \mathbb{Z}_n$, compute $\hat{m} = r^e \cdot m (\mathrm{mod} n)$. Obtain $\hat{\sigma}$ s.t. $(\hat{\sigma})^e = \hat{m} (\mathrm{mod}\ n)$.
    2. Have $\sigma = \frac{\hat{\sigma}}{r}$ Existential Forgery: (σ, m) since $\sigma^e = (\hat{\sigma}/r)^e = \frac{r^e \cdot m^e}{r^e} (\mathrm{mod} n) = m^e (\mathrm{mod} n)$.

9. **Standard Signatures** PKCS 1.5 which is not full domain. Problems with provability and thus no security analysis.

10. **Hash-based Signatures** - By virtue of no factorization, we have that these are post-quantum secure.

    - The basic scheme - Lampart for $v = 256$ bits. Let $H : X \to Y$ be a one-way hash func. Uses SHA-256.

11. Building many-time signatures using Lampart.
    1. Generate n one-time (pk, sk) key pairs.
    2. Build Merkle tree from n PKs.
    3. Set pk as root of Merkle tree. Let sk just be the secret keys be generated.
    4. Signer needs to store state of what key the last message was signed with. Can NEVER sign two messages with same key.



12. Public Key Management: **Certificates**

    - Certificates bind a public key to an identity. Revocation methods: expiration, CRLSet.

    - General Structure Certificate: issuer name, issuer subject, pk, validity period, CA's signature on pre-cert.

    - Issuing wrong certificates can lead identity misbinding attacks

10 Authenticated Key Exchange

1. Key exhcange in the presence of an active adeversary:

   - An adversary has total control over entire network i.e. can modify, inject, and delete packets.

   - Goal: Maintain Forward Secrecy, HSM (Hardware Security Module) Security, Static security.

   - HSM Security: n queries to HSM should only make n sessions insecure. HSM stores all private stuff for an exchange

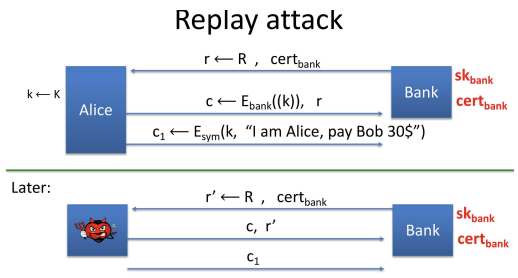
2. Building One-side AKE Protocols:

   - $\text{cert}_{\text{bank}}$ contains pk_{bank} implies that Bank has sk_{bank}.

   - $E_{\text{bank}}\left((m,r)\right) = E\left(pk_{\text{bank}}\left(m,r\right)\right)$ where E is chosen-ciphertext secure (CCA secure).

   - $S_{\text{alice}}\left((m,r)\right) = S\left(k_{\text{alice}}\left(m,r\right)\right)$ where S is a secure signing alg.

   - Open to replay attacks when r is not encrypted:



3. Building Two-Side AKE protocols

   - Alice has $sk_{\text{alice}}, \text{cert}_{\text{alice}}$ and Bank has $sk_{\text{bank}}, \text{cert}_{\text{bank}}$.

   - Bank sends over randomness r and $\text{cert}_{\text{bank}}$. Alice generates $k \in K$ and obtains $c \leftarrow E_{\text{bank}}((k, \text{"alice"}))$. Also generates $\sigma \leftarrow S_{\text{alice}}((r, c, \text{"bank"})), \text{cert}_{\text{alice}}$.

   - Alice sends over c, σ to Bank.

   - Insecure variant: when r is encrypted instead of "alice", Alice's identity. Prone to **identity misbinding attacks**.

4. Both one-sided AKE and two-sided AKE right now don't have forward secrecy given above constructions.

   - If adversary obtains sk_{bank} at some point in communication, then all communications are insecure forever.

5. Making One-sided AKE forward secret:

   - Bank generates a new pair (sk, pk) every time it initiates a session with Alice. Sends over $pk, \text{cert}_{\text{bank}}$ along with $\sigma \leftarrow S_{\text{bank}}((pk))$. Alice, sends over $c \leftarrow E(pk, k)$.

   Deletes sk after shared key is exchanged i.e. (pk, sk) are ephemeral.

   - Insecure variant: Not signing pk. Adversary launches person in the middle attack.

     - Intercepts $pk, \text{cert}_{\text{bank}}$. Instead sends over pk' from $\text{Gen}_{\text{adv}} \leftarrow (pk', sk')$.

     - Adv. can now get shared key since $c \leftarrow E(pk', k)$. Can be decrypted using Adv's secret key.

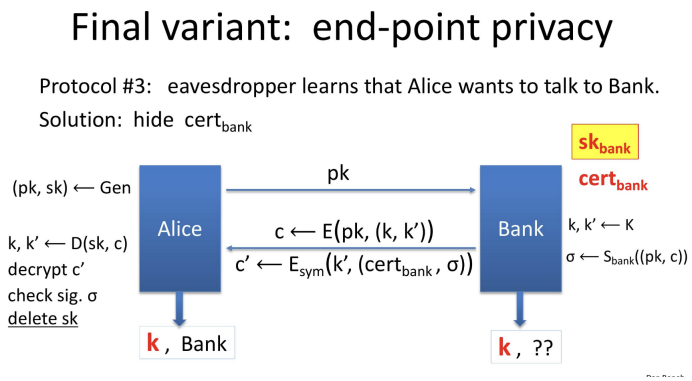
5. **Achieving HSM security**

   - Alice generates $(pk, sk) \leftarrow Gen()$. Sends over pk. Bank generates $k \in K$.

   - Bank sends over $c \leftarrow E(pk, k)$, $\text{cert}_{\text{bank}}$, $\sigma \leftarrow S_{\text{bank}}((pk, c))$.

   - Alice deletes sk after doing $D(sk, c)$ and cheking σ. Main point: HSM needed to sign ephemeral pk from client.

6. Final variant: End Point Privacy:



   - Goal is to hide Bank's identity from eavesdroppers. **Solution** encrypt σ and $\text{cert}_{\text{bank}}$ using keys generated $k, k' \leftarrow K$.

11 Identification Protocols

1. $360,000,000$ words cover 25% passwords. Usual attacks come in the form of dictionary attacks where attacker fixes password and tries a bunch of usernames. Usually success comes after ≈ 33 tries. Can be mitigated using IP-rate limiting.

2. **Offline Dictionary attack:**

   1. Attacker obtains hashed passwords. from server. Very fast due to hashing algorithms

   2. Scanning through $360,000,000$ guesses can recover about 23% of passwords. Sort hashed words and obtained list and then compare: $O(|\text{Dict}| + |\text{F}|)$.

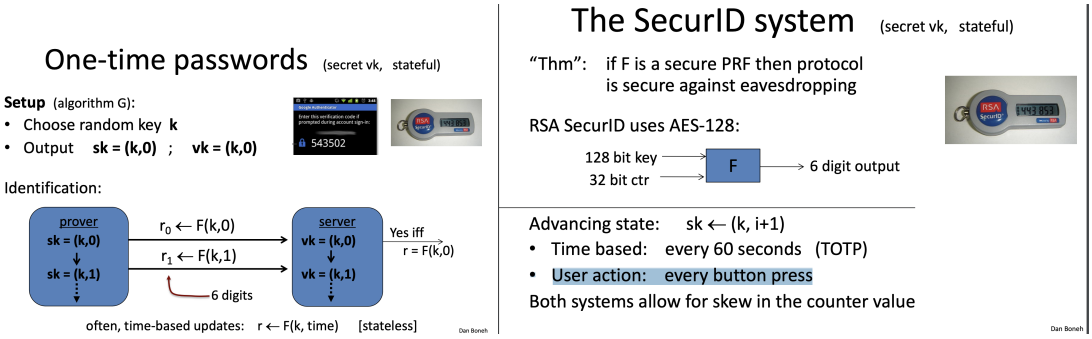
   3. To prevent additive running time. Use salt: 64 bits. Attacker has to re-compute all hashes for every user it wishes to attack.

3. To hash passwords:

   - Use a keyed hash function like HMAC where key is stored in HSM.

   - Use a **slow, space-hard** function. Example: scrypt

4. **One-time Passwords**



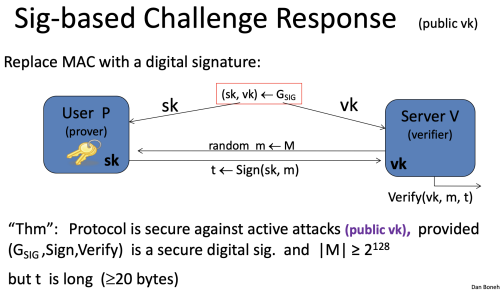
- Thm: If F is a secure PRF, then above protocol is secure against eavesdropping.

5. **The S/Key System**
   1. Prover chooses random key $sk = (k, i)$, sends $t \leftarrow H^{(i)}(k)$. Sets $sk = (k, i-1)$ for next time.
   2. Verifier has $vk = H^{(i+1)}(k)$ and t. Checks if $H(t) = vk$.

   - Allows for vk to be public since H is considered to be one-way on iterates.

6. Active Attacks in Identification:

   - Signature Based Systems



12 Zero-Knowledge Proofs

1. **Class NP**:

   - L $\subseteq \{0,1\}^n$ is in NP if $\exists w \in \{0,1\}^n$ s.t. $M(x, w) = 1$
     - x is called "statements"
     - w is called "witnesses"
     - "example": $\alpha \in Z_q$ is a witness that $(g, h, \mu, v) \in L_{EDL}$.

2. **Zero Knowledge Proof Systems**:

   - Pair of probabilistic polynomial time algorithms for prover, verifier (P, V) s.t. after interactions between: $P(x, w)$ and $V(w)$, V outputs yes/no.
   - (1). "complete": is $M(x, w) = 1$, i.e. $x \in L$ then $Pr[P(x, w) \Leftrightarrow V(w) = \text{yes}] = 1$.
   - (2). "sound": $\forall x \notin L$: (cheating prover cannot convince our verfidier that $x \in L$):
     - $\forall \hat{p} : \Pr[(\hat{P}(x) \leftrightarrow v(x)) = yes] \leqslant \text{neg}$

3. **Honest Verifier Zero Knowledge** (HVZK)

- Protocol reveals nothing to V other than $x \in L$.

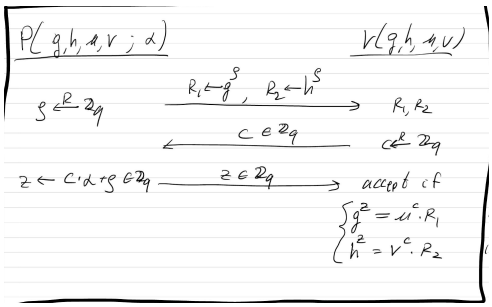
- For x, w let transcript $(P(x,w) \Leftrightarrow v(x))$ be the seq. of messages between $P(x,w)$ and $V(x)$ (a distribution).

- (P, r) is HVZK for L:

- if $\exists S$ (simulator) s.t. $\forall x \in L$: the distributions $\{S(x)\}$ and $\{\text{transcript}(p(x,w) \leftrightarrow r(x))\}$ are completely indistinguishable.

- The: every $L \in$ NP has an "eff" ZK proof system.

- Proof of Knowledge System for L_{EDL}:



- Observe that the probability a malicious prover will be as follows given that it sends over $u = g^\alpha$ and $v = h^\beta$ where $\alpha \neq \beta$ and transcript is $(R_1 = g^\alpha, R_2 = h^\beta, c, z)$:

$$Pr[\text{V accepts}] = Pr[z = \alpha c + \rho_1 \text{ and } z = \beta c + \rho_2] = Pr[c = \frac{\rho_1 - \rho_2}{\beta - \alpha}] = \frac{1}{q}, \text{"neg"}$$

- Simulator Algorithm for L_{EDL} HVZK system:

- Sim(g, h, u, v):

- $\left[\begin{array}{l} \text{choose: } \quad c, z \xleftarrow{r} Z_q \\ \text{set } R_1 = g^z/u^c, R_2 = h^z/v^c \\ \text{output } (R_1, R_2, c, z) \end{array}\right]$

- Observe that $R_1 = g^\rho, R_2 = h^\rho$ where $\rho = z - c\alpha \in Z_q$.

- So c is uniform in Z_q and z is s.t. (1) and (2) from above hold.

(Signature from DLog)

Schnorr sigs: [public coin ZKPK ⇒ sig. scheme]

need: $H: M \times G \to \mathbb{Z}_q$

$\text{Gen}()$: $\alpha \xleftarrow{R} \mathbb{Z}_q$, $h \leftarrow g^\alpha \in G$

$sk := \alpha$, $pk = h$

$S(sk=\alpha, m)$: $\rho \xleftarrow{R} \mathbb{Z}_q$, $R \leftarrow g^\rho \in G$ ← 1st msg from P

$c \leftarrow H(m, R) \in \mathbb{Z}_q$ ← chall. from V

$z \leftarrow \rho + c \cdot \alpha \in \mathbb{Z}_q$ ← 2nd msg from P

output $\sigma = (c, z)$ ← 64 bytes. (much shorter than RSA (256 bytes))

$V(PK=h, m, \sigma=(c,z))$: accept if

$H(m, \underbrace{g^z/h^c}_{R}) \stackrel{?}{=} c$

$[g^z = h^c \cdot R]$

Thm: (Gen, S, V) is secure assuming DLOG hard in G and H is modeled as a random oracle.