1. Private Key Encryption Scheme: Defined by message space M and algorithms.
   (a) *Gen*: Outputs $k \in K$
   (b) *Enc*: Outputs $c \leftarrow Enc_k(m)$
   (c) *Dec*: Outputs $m \leftarrow Dec_k(c)$ or "error"

   For all $m \in M$ and k from *Gen*, $Dec_k(Enc_k(m)) = m$.

2. Shift Cipher:
   (a) M={String over lowercase alphabet}
   (b) *Gen*: Chooses uniform $k \in \{0, \ldots, 25\}$
   (c) $Enc_k(m_1 \ldots m_t)$: For each m_i, output $c_1 \ldots c_t$, where

$$c_i := [m_i + k] \mod 26$$

   (d) $Dec_k(c_1 \ldots c_t)$: For each c_i, output $c_1 \ldots c_t$, where

$$m_i := [c_i - k] \mod 26$$

3. Byte-Wise Shift Cipher:
   (a) M={String of bytes}
   (b) *Gen*: Chooses uniform $k \in \{0x00, \ldots, 0xFF\}$ (Gives 256 Possible Keys)
   (c) $Enc_k(m_1 \ldots m_t)$: For each m_i, output $c_1 \ldots c_t$, where

$$c_i := m_i \oplus k$$

   (d) $Dec_k(c_1 \ldots c_t)$: For each c_i, output $c_1 \ldots c_t$, where

$$m_i := c_i \oplus k$$

4. Vigenere Cipher: Make the key a string, where every character is shifted by the amount dictated by the next character of the key. We wrap around the key as more characters are needed. Decryption reverses the process.

   (a) Attack on a fixed key length: Assume a 14-character key. Look at every 14th character of the ciphertext, group them all into separate streams.

      i. Let $p_i (0 \leq i \leq 25)$ denote the frequency of the $i-th$ English character.
      $$\sum_i p_i^2 \approx 0.065$$

      ii. If q_i is the observed frequency of the $i-th$ character in a ciphertext stream, we expect that if the stream is j, we have $q_{i+j} \approx p_i$ for all i.
      $$\sum_i p_i q_{i+j} \approx 0.065$$

      iii. Test for every value j, take the highest sum.

   (b) Find the key length:

      i. When using proper key length, the ciphertext frequencies of a given stream $\{q_i\}$ will be shifted versions of $\{p_i\}$.
      $$\sum q_i^2 \approx \sum p_i^2 \approx 0.065$$

      ii. With incorrect key lengths, ciphertext letters will be uniform.
      $$\sum q_i^2 \approx \sum \left(\frac{1}{26}\right)^2 \approx 0.038$$

      iii. Thus, we find the key length N to maximize
      $$\sum q_i^2$$

5. Byte-wise Vigenere Cipher: The key is a string of bytes, the plaintext is a string of bytes. To encrypt and decrypt, $\oplus$ each character in the text with the next character of the key. Wrap around key as needed.

   (a) Determine Key Length: Let p_i be the frequency of a byte i in ASCII.

      i. For a candidate key length, take the first stream q and calculate $\sum q_i^2$. We must find the sum satisfying the following. The correct key length yields a large value for all streams.

$$\sum q_i^2 \approx \sum p_i^2$$

   (b) Determine the i-th Key Byte: Assume key length known.

      i. Look at i-th ciphertext stream, try decrypting with every byte value b. We must find something to satisfy the following, where p_i' corresponds to English-letter frequencies.

$$\sum q_i' p_i' \approx \sum p_i'^2 \approx 0.065$$

   /P

6. Perfect Secrecy: An encryption scheme with message space M and ciphertext space C is perfectly secret if for every distribution over M and every $m \in M$ and every $c \in C$ with $\Pr[C = c] > 0$, the following holds.

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

   (a) Proof: If (Gen, Enc, Dec) is a perfectly secret, then $|K| > |M|$

      i. Take the uniform distribution on M and any ciphertext c. Consider $M(c) = \{Dec_k(c)\}_{k \in K}$.

      ii. Since $|M(c)| \leq |K| \leq |M|$, there is some m not in $M(c)$

      iii. $\Pr[M = m | C = c] = 0 \neq Pr[M = m]$

7. One Time Pad: Perfectly secret scheme.
   (a) $M = \{0,1\}^n$
   (b) Gen : Choose a uniform $k \in \{0,1\}^n$
   (c) $Enc_k(m) = k \oplus m$
   (d) $Dec_k(c) = k \oplus c$
   (e) Correctness: $Dec_k(Enc_k(m)) = k \oplus (k \oplus m) = m$.
   (f) Perfect Secrecy: Fix an arbitrary distribution over $M = \{0,1\}^n$, and an arbitrary $m, c \in \{0,1\}^n$.

$$\Pr[C = c] = \sum_{m'} \Pr[C = c | M = m'] * \Pr[M = m']$$

$$= \sum_{m'} \Pr[k = m' \oplus c | M = m'] * \Pr[M = m']$$

$$= \sum_{m'} 2^{-n} * \Pr[M = m'] = 2^{-n}$$

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

$$= \frac{\Pr[k = m \oplus c | M = m] * \Pr[M = m]}{2^{-n}} = \frac{2^{-n} * \Pr[M = m]}{2^{-n}}$$

$$= \Pr[M = m]$$

   Thus, this scheme is perfectly secret.
   (g) Cracking One Time Pad with Multiple Messages:
      i. Given $c_1 = k \oplus m_1$ and $c_2 = k \oplus m_2$, we can get $c_1 \oplus c_2 = m_1 \oplus m_2$.
      ii. Since letters start with 01 and spaces begin with 02, the $\oplus$ of two letters or two spaces gives 00 while the $\oplus$ or a letter and a space gives 01. Since spaces are more unlikely than letters, given many messages, if $m_i \oplus m_j$ likely produces 01 for some fixed m_i, then m_i is likely a space character.

8. Indistinguishability Experiment: Given a scheme $\Pi = (Gen, Enc, Dec)$ over message space M.

   (a) Design a random experiment $PrivK_{A,\Pi}$

      i. Two encrypted messages m_0, m_1 are made. For some $b \in (0, 1)$, we set $c_b \leftarrow Enc_k(m_b)$. We send m_0, m_1, c_b to the adversary.

      ii. A returns 0 or 1 based on if the plaintext was m_0 or m_1 in $b' \leftarrow A(c)$.

      iii. A succeeds if $b = b'$, in which case $PrivK_{A,\Pi} = 1$. Otherwise, the experiment evaluates to 0.

   (b) Π is perfectly indistinguishable for all adversaries A if the following holds.

$$\Pr[PrivK_{A,\Pi} = 1] = \frac{1}{2}$$

      i. Π is perfectly indistinguishable if and only if Π is perfectly secret.

   (c) Concrete Computational Indistinguishability: We define (t, ϵ) indistinguishability as indistinguishability where security may fail with probability $\leq \epsilon$ when restricted to time $\leq t$ CPU cycles. Π is (t, ϵ)-indistinguishable if for all attackers A running in time at most t, the following holds.

$$\Pr[PrivK_{A,\Pi}] \leq \frac{1}{2} + \epsilon$$

   (d) Asymptotic Computational Indistinguishability: Security may fail with probability $negl(n)$ for polynomial time attackers.

      i. $negl(n) : f$ is negligible if for every polynomial p, it holds that $f(n) < \frac{1}{p(n)}$ for a large enough n. Or, it decays faster than an inverse polynomial.

   Π is computationally indistinguishable for all probabalistic polynomial time attackers A if there is a negligible function ϵ such that the following holds.

$$\Pr[PrivK_{A,\Pi}] = \frac{1}{2} + \epsilon(n)$$

9. Pseudorandomness: Informally, a distribution on n-bit strings is pseudorandom if it can not be differentiated from a completely uniform distribution.

   (a) Concrete Pseudorandomness: If D is a distribution on p-bit strings, D is (t, ϵ)-pseudorandom if for all A running in time at most t, the following holds. The adversary is able to sample from the distribution.

$$\left| \Pr_{x \leftarrow D}[A(x) = 1] - \Pr_{x \leftarrow U_p}[A(x) = 1] \right| \leq \epsilon$$

   (b) Asymptotic Pseudorandomness: If D is a distribution over $p(n)$-bit strings, pseudorandomness is a porperty of a sequence of distributions. We say that $\{D_n\} = \{D_1, D_2, \ldots\}$ is pseudorandom if for all probabilistic polynomial distinguishers A, there is a negligible function ϵ such that the following holds.

$$\left| \Pr_{x \leftarrow D_n}[A(x) = 1] - \Pr_{x \leftarrow U_{p(n)}}[A(x) = 1] \right| \leq \epsilon(n)$$

10. Pseudorandom Generator: Let G be a deterministic polynomial time algorithm that is expanding ($|G(x)| = p(|x|) > |x|$). G defines a sequence of distributions $\{D_n\}$ of outputs given a uniform input into x. We say that G is pseudorandom if and only if $\{D_n\}$ is pseudorandom if and only if for all efficient distinguishers A able to query for outputs, the following holds.

$$\left| \Pr_{x \leftarrow U_n}[A(G(x)) = 1] - \Pr_{y \leftarrow U_{p(n)}}[A(y) = 1] \right| \leq \epsilon(n)$$

11. Psuedo One-Time Pad:

    (a) Let G be a deterministic algorithm, with $|G(k)| = p(|k|)$

    (b) $Gen(1^n)$: Outputs uniform n-bit key k.

    (c) $Enc_k(m)$: Output $G(k) \oplus m$

    (d) $Dec_k(c)$: Output $G(k) \oplus c$

    To prove this is computationally secret, assume there is an attacker A that breaks this scheme and use it to build an efficient D breaking the pseudorandomness of G.

    (e) Proof of Computational Secrecy

$$\text{Let } \mu(n) = \Pr[PrivK_{A,\Pi}(n) = 1]$$

$$\Pr_{x \leftarrow U_n}[D(G(x)) = 1] = \mu(n)$$

$$\Pr_{y \leftarrow U_{p(n)}}[D(y) = 1] = \frac{1}{2}$$

$$|\mu(n) - \frac{1}{2}| \leq negl(n)$$

$$\Pr[PrivK_{A,\Pi}] \leq \frac{1}{2} + negl(n)$$

12. Multiple Message Security: Fix some Π, A. We will define a randomized experiment $PrivK^{mult}_{A,\Pi}(n)$ as follows.

    (a) Consider two vectors $(m_{0,1}, \ldots, m_{0,t})$ and $(m_{1,1}, \ldots, m_{1,t})$ with $|m_{0,i}| = |m_{1,i}|$ for all i.

    (b) For some $k \leftarrow Gen(1^n)$ and $b \leftarrow \{0,1\}$, for all i set $c_i \leftarrow Enc_k(m_{(b,i)}$ in vector $(c_1, \ldots, c_t)$.

    (c) Send the ciphertext vector and both message vectors to the adversary. The adversary returns b' and succeeds if $b' = b$, in which case the experiment evaluate to 1.

    We say that Π is multiple-message indistinguishable if the following holds for some neglibile function ϵ.

$$\Pr[PrivK^{mult}_{A,\Pi} = 1] \leq \frac{1}{2} + \epsilon(n)$$

13. Chosen Plaintext Attack Security: Fix some Π, A. We will define a randomized experiment $PrivK^{CPA}_{A,\Pi}$ as follows.

    (a) $k \leftarrow Gen(1^n)$

    (b) A^{1^n} interacts with an encryption oracle $Enc_k()$, and is given m_0, m_1 of the same length.

    (c) We set $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$, and then give c to A.

    (d) A continues to interact with the oracle

    (e) A outputs b', succeeding if $b = b'$, in which case the experiment evaluate to 1.

    We say Π is chosen-plaintext attack secure if for all probabalistic polynomial time attacks A, there is a negligible function ϵ such that the following holds.

    $$\Pr[PrivK^{CPA}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$

    Note: CPA-security is stronger than multiple-message indistinguishability. As a corollary, if Π is CPA-secure, it is also multiple-message indistinguishable.

    Note that no deterministic and stateless encryption scheme satisfies this

14. Uniform Functions: Where $Func_n$ is all functions mapping from $\{0,1\}^n$ to $\{0,1\}^n$.

    (a) Size of a function in $Func_n$: $n2^n$ bits

    (b) $|Func_n| = 2^{n2^n}$

15. Pseudorandom Function: Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, deterministic algorithm. We will define $F_k(x) = F(k,x)$, where the first input is the key.

    (a) We will assume F is length-preserving, or $|F(k,x)| = |k| = |x|$.

    (b) Choosing a uniform $k \in \{0,1\}^n$ is equivalent to choosing a random function $F_k : \{0,1\}^n \to \{0,1\}^n$. Note that there are at most 2^n possible functions F_k.

    We say that F is a pseudorandom function if for all polynomial time distinguishers D having oracle access to its given function and some neglibile function ϵ, the following holds.

    $$\left| \Pr_{k \leftarrow \{0,1\}^n}[D^{F_k} = 1] - \Pr_{f \leftarrow Func_n}[D^{f()} = 1] \right| \leq \epsilon(n)$$

    Note that a PRF F immediately implies a PRG G.

16. Pseudorandom Permutation: We say F is a permutation if it is a bijective function. Note that for a large enough n, a random permutation is indistinguishable from a random function. Similar to a pseudorandom function, the following must hold for a function to be a pseudorandom permutation, but F_k is a pseudorandom permutation and f is a permutation.

    $$\left| \Pr_{k \leftarrow \{0,1\}^n}[D^{F_k} = 1] - \Pr_{f \leftarrow Func_n}[D^{f()} = 1] \right| \leq \epsilon(n)$$

    Note that a block cipher is a practical construction of a pseudorandom permutation.

17. CPA-Secure Encryption Scheme: Let F be a length-preserving, keyed function.

    (a) $Gen(1^n)$: Choose a uniform key $k \in \{0,1\}^n$.

    (b) $Enc_k(m)$: Let $|m| = |k| = n$. Choose a uniform $r \in \{0,1\}^n$ and output ciphertext $< r, F_k(r) \oplus m >$.

    (c) $Dec_k(m)$: Outputs $c_2 \oplus F_k(c_1)$.

    If F is a pseudorandom function, then this scheme is CPA-secure.

18. CTR-Mode Encryption: Proven to be pseudorandom if F is pseudorandom.
    (a) $c_0 \leftarrow \{0,1\}^n$ selected at random
    (b) For $i = 1$ to t, do $c_i = m_i \oplus F_k(ctr + i)$
    (c) Output $c_0, c_1, \ldots, c_t$
19. CBC-Mode Encryption: Proven to be pseudorandom if F is pseudorandom.
    (a) $c_0 \leftarrow \{0,1\}^n$ selected at random
    (b) For $i = 1$ to t, do $c_i = F_k(m_i \oplus c_{i-1})$
    (c) Outputs $c_0, c_1, \ldots, c_t$.
    (d) $Dec_k(c)$ requires F to be invertible.
20. Stream Ciphers: Infinite seed of psuedorandom bits on demand. They are secure if the output stream is pseudorandom. Has synchronized and unsynchronized modes of operation.
    (a) Synchronized: Sender and receiver maintain states. Makes sense when two parties are commuicating online and receiving messages in order
    (b) Unsynchronized: States are not maintained. Generally, you have to send a key over if two parties are communicating.
21. Message Authentication Code: Defined by three probabilistic polynomial time algorithms.
    (a) Gen: Takes input 1^n, outputs k. Assume $|k| \geq n$.
    (b) Mac: Takes input k and message, outputs tag t. $t \leftarrow Mac_k(m)$.
    (c) Vrfy: Takes k, m, and t, and outputs 1 if the tag and message match. Otherwise, 0.

    For all m and k, we must have $Vrfy_k(m, Mac_k(m)) = 1$ in a correct scheme.

22. Existential Unforgeability: We fix A, Π and define the following random experiment $Forge_{A,\Pi}(n)$.

    (a) $k \leftarrow Gen(1^n)$

    (b) A interacts with an oracle $Mac_k()$. Let M be the set of messages submitted to this oracle.

    (c) A outputs (m, t).

    (d) A succeeds and the experiment evaluates to 1 if $Vrfy(m, t) = 1$ and $m \notin M$.

    We say that Π is secure, or existentially unforgeable if the following holds for some negligible ϵ.

    $$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n)$$

23. Fixed-Length Secure MAC Scheme: Let F be a length-preserving pseudorandom function, or block cipher.

    (a) Gen: Choose a uniform k for F.

    (b) Mac_k: Output $F_k(m)$

    (c) $Vrfy_k(m, t)$: Output 1 iff $F_k(m) = t$

24. Secure CBC-MAC: Prepend the length of a message to the message.

    (a) Gen: Choose a uniform k for F.

    (b) Mac_k: First, start with message length l.

    $$t = F_k(m_l \oplus F_k(\ldots \oplus F_k(m_1 \oplus F_k(l))))$$

    (c) $Vrfy_k(m, t)$: Output 1 is the above holds for some m, t.

25. Chosen Ciphertext Attack Security: Fix some Π, A. We will define a randomized experiment $PrivK^{CCA}_{A,\Pi}$ as follows.

    (a) $k \leftarrow Gen(1^n)$

    (b) A^{1^n} interacts with an encryption oracle $Enc_k()$ and decryption oracle $Dec_k()$, and is given m_0, m_1 of the same length.

    (c) We set $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$, and then give c to A.

    (d) A continues to interact with both oracles, but can not request decryption of c.

    (e) A outputs b', succeeding if $b = b'$, in which case the experiment evaluate to 1.

    We say Π is chosen-ciphertext attack secure if for all probabalistic polynomial time attacks A, there is a negligible function ϵ such that the following holds.

$$\Pr[PrivK^{CCA}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$

26. Malleability: A scheme is malleable if it is possible to modify a ciphertext and thereby cause a predictable change in the plaintext.

    CCA-security implies non-malleability.

27. Padding Oracle Attack: Start altering message from the start until you infringe on the padding, in which case an error will be returned, if a change on the i-th ciphertext bit only changes the i-th bit of the encoded data. Then, keep altering the ciphertext until you guess a value that correctly matches the padding, in which case you may be able to attain a key for that bit.

28. Authenticated Encryption: We use CPA-Secure encryption schemes and MAC-secure encryption schemes in combination to create CCA-Secure Schemes. These schemes achieve both CCA-Security and Existential Unforgeability.

    (a) Encrypt then Authenticate: Works, as encrypted, invisible data is authenticated, meaning it will not be altered.

    (b) Encrypt and Authenticate: Does not work, because if you send MAC encryption and CCA-Secure encryption simultaneously, the MAC may not be CCA-Secure and the CCA may not be unforgeably.

    (c) Authenticate, then Encrypt: Padding Oracle Attack still works, since an adversary can still alter a message and get an error during encryption.

29. Secure Sessions: Consider parties wishing to communicate securely and with integrity over a session. We must consider possible attacks to the session.

    (a) Replay Attack: Resend an intercepted message again. May for instance, result in multiple charges for one bank request.

    (b) Reordering Attack: Messages are mixed up.

    (c) Reflection Attack: Messages sent out are redirected to sender.

    Secure sessions must use counters to keep track of message numbers and order as well as identifiers to identify who is sending the message as part of the data to thwart these attacks.

30. Cryptographic Hash Function: A deterministic function mapping arbitrary length inputs to a short, fixed-length output. Generally keyed, though we assume unkeyed for simplicity.

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

31. Collision Resistance: Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash funciton. A collision is a distinct pair of inputs x, x' such that $H(x) = H'(x)$. H is collision resistant if finding a collision in H is computationally infeasible.

32. Birthday Attack: Compute $H(x_1), \ldots, H(x_k)$. What is the probability of a collision as a function of k?

$$\Pr[\text{Find a collision}] = \frac{1}{N} + \frac{2}{N} + \ldots + \frac{k-1}{N} = \frac{k(k-1)}{2N} = \frac{k^2}{2N} - \frac{k}{2N}$$

Now, let $k = \sqrt{N}$. We get the following.

$$\Pr[\text{Find a collision}] = \frac{1}{2} - \frac{1}{2\sqrt{N}} \approx \frac{1}{2}$$

If you have a hash mapping to n bits, the security level is therefore $\frac{n}{2}$ bits.

33. Merkle-Damgard Transform: Create a hash function H for a sequence of messages from a hash function h for one message. Let $IV \leftarrow \{0,1\}^n$.

$$H(m_1, \ldots, m_B) = h(h(h(h(IV, m_1), m_2), \ldots, m_B), B+1)$$

Claim: H is collision resistant if h is collision resistant.

Suppose we find a collision for H. Let $k = h(h(h(IV, m_1), m_2), \ldots, m_B)$, and $k' = k = h(h(h(IV, m_1), m_2), \ldots, m_B)$. Then, we have found some $h(k, B+1) = h(k', B+1)$. If $k \neq k$, then we are done. Otherwise, we can find another k''.

34. Hash-and-Mac: A secure MAC for long messages.

    (a) Gen' : Find some H by Merkel-Damgard, take Gen

    (b) $Mac'_k : t \leftarrow Mac_k(H(m))$

    (c) $Vrfy'_k(m, t)$: Check $Vrfy(t, H(m))$

35. Outsourced Storage Problem: Lets say you have multiple files you want to store on another server, and need a way to verify/update these files.

    (a) Hash-All: The client stores $h_1 = H(f_1), \ldots, h_n = H(f_n)$. We use hash functions to verify if each file sent back is $h_1, \ldots, h_n$ when we request one.

        i. Communication (Bits to Download): 0
        ii. Update: 0
        iii. Local Files: 256*n

    (b) Hash-All: The client stores $h_1 = H(f_1), \ldots, h_n = H(f_n)$. We use hash functions to verify if each file sent back is $h_1, \ldots, h_n$ when we request one.

        i. Communication (Bits to Download): $O(n)$ (We need all files to hash!)
        ii. Update: $O(n)$
        iii. Local Files: $O(1)$

    (c) Hash-Prefix: We verify with $h(h(h(f_1, f_2), f_3), \ldots)$.

        i. Communication (Bits to Download): $*O(n)$
        ii. Update: $*O(n)$
        iii. Local Files: $O(1)$

        Note that for certain files, $*$ is far more efficient.

36. Number Theory Essentials

(a) Primes: Natural number x whose only divisors are x and 1. There are $\approx \frac{n}{\log(n)}$ primes less than n.

(b) GCD: $d = \gcd(a, b)$ is the smallest positive integer satisfying $d = ax + by$ for variables i,j.

(c) Euclidean Algorithm: Note that $\gcd(a, b) = \gcd(b, a \mod b)$. Then, we can find $gcd(a, b)$ with the following algorithm.

```
function EUCLIDEAN(a, b)
    if b=0 then
        return a
    else
        return Euclidean(b, a mod b)
    end if
end function
```

(d) Group: Set G with an operation $\cdot$ that satisfies the following group axioms.

i. Closure under $\cdot$

ii. Associativity under $\cdot$

iii. There exists an Identity Element

iv. Every element has an inverse under $\cdot$ We say that a group is Abelian if it also satisfies commutativity.

(e) Group Order: Number of elements in a group

(f) $\mathbb{Z}_N^*$: Let $N = pq$, where $p, q \in \mathbb{P}$. Then, $|Z_N^*| = \Phi(p)\Phi(q) = (p-1)(q-1)$.

$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N^{**} \colon \gcd(x, N) = 1\}$$

$$\mathbb{Z}_N^{**} = \{1, \ldots, N-1\},\ N \in \mathbb{P}$$

(g) If G is a group of order m and $\gcd(e, m) = 1$, then g^e is a permutation.

(h) Cyclic Group: If we have a group G of order m and there is an element g of order m, then g is a generator and G is cyclic.

(i) Chinese Remainder Theorem: If we have $x \mod p$ and $y \mod q$ for some coprime p, q, then we can produce some a such that $a \equiv x \mod p$ and $a \equiv y \mod q$.

(j) Invertibility: b is invertible mod N if $\gcd(b, N) = 1$.

(k) Fermat's Little Theorem: For any element g of group G of order m, it holds that $g^m \equiv 1 \mod N$.

i. Corollary I: $g^x = g^{x \mod m}$

ii. Corollary II: If $d \equiv e^{-1} \mod m$, then g^d is the inverse of g^e. (We then have $g^{de} \equiv g^1 \mod N$)

iii. Corollary III: If $\gcd(e, m) = 1$, then $x^e \mod N$ is a permutation.

37. Efficient Modular Exponentiation: Efficiently find $a^b \mod N$. Instead of doing $a * \ldots * a$, do $(((a^2)^2)\ldots)^2$.

38. Discrete Log Assumption: Let G be a cyclic group of order q and let g be a generator of g. Let h be a group element. It is very difficult to find an x such that $h = g^x$

Let G be a group generation algorithm that on input 1^n, outputs a cyclic group G with order q such that $||q|| = n$ and a generator g. The experiment $Dlog_{A,G}(n)$ is as follows.

(a) Compute $(G, q, g) \leftarrow G(1^n)$.

(b) Choose a uniform $h \in G$

(c) Run $A(G, q, g, h)$ to get x

(d) Experiment evaluates to 1 is $g^x = h$.

We say that the discrete-logarithm problme is hard relative to G if for all PPT algorithms A, the following holds.

$$Pr[Dlog_{A,G}(n) = 1] \leq negl(n)$$

39. RSA/Factoring Assumption: Given $N = pq$ where $p, q \in \mathbb{P}$, it is very difficult to find out p and q. Note that $\mathbb{Z}_N^*$ is the RSA group.

    Let $GenRSA$ be a PPT algorithm that given an input 1^n, outputs a modulo N that is the product of two $n-$bit primes and also integers $e, d > 0$ with $\gcd(e, \phi(N)) = 1$ and $ed \equiv 1 \mod \phi(N)$. The RSA experiment $RSA - inv_{\mathcal{A},GenRSA}(n)$ is as follows.

    (a) Run $GenRSA(1^n)$ to obtain (N, e, d)

    (b) Choose a uniform $y \in \mathbb{Z}_N^*$

    (c) $\mathcal{A}$ is given (N, e, y) and outputs $x \in \mathbb{Z}_N^*$

    (d) The output is 1 if $x^e = y \mod N$, 0 otherwise.

    We say that the RSA problem is hard relative to GenRSA if for all PPT algorithms $\mathcal{A}$, there exists a negligible function $negl$ such that $Pr[RSA - inv_{\mathcal{A},GenRSA}(n) = 1] \leq negl(n)$.

40. Diffie-Hellman Problems: Fix a cyclic group G and a generator g. We define $DH_g(h_1, h_2) = DH_g(g^x, g^y) = g^{xy}$

    (a) Computational Diffie-Hellman Problem: Given g, h_1, h_2, compute $DH_g(h_1, h_2)$.

    (b) Decisional Diffie-Hellman Problem: Given g, h_1, h_2, distinguish $DH_g(h_1, h_2)$ from a uniform element of G.

    We say that the DDH problem is hard relative to G if for all PPT algorithms A, the following holds.

    $$\Pr[A(G, g, q, g^x, g^y, g^{xy}) = 1] - \Pr[A(G, q, g, g^x, g^y, g^z) = 1] \leq negl(n)$$

    Theorem: If the discrete-logarithm problem is easy, then so is CDH. If the CDH problem is easy, so is the DDH problem.

41. Group Selection: The discrete logarithm problem is not hard in all groups. For cryptographic applications, it is best to use prime order groups, since the discrete log problem is easier if the order of the group has small primes. The following groups are currently suitable for use.

    (a) Let $p, q \in \mathbb{P}$. We select $p = tq + 1$. Then, $\mathbb{Z}_p^*$ has order $p - 1 = tq$. Then, we define the group as the subgroup of t^{th} powers.

    $$G = [x^t \mod p] | x \in \mathbb{Z}_{\text{ı}}^*]$$

    (b) Elliptic Curve Groups

42. Key Distribution Problem: How do you establish a secure channel between two members who have no prior communications? (Motivation for Private Key Cryptography)

    (a) New Directions in Cryptography by Diffie and Hellman: Creates 'Assymetric' or Private Key Encryption-related key exhange protocols.

43. Key Exchange Protocol Security: For key exchange protocol Π, we devise the experiment $KE_{A,\Pi}$.

    (a) Honest parties run Π with security parameter n, resulting in transcript $trans$ and shared key k.

    (b) We choose uniform bit b. If $b = 0$, then set $k' = k$. If $b = 1$, then choose uniform $k' \in \{0,1\}^n$.

    (c) $KE_{A,\Pi}$ evaluates to 1, or A suceeds if $b' = b$.

    We say that Π is secure against passive eavesdropping if for all PPT adversaries A, the following holds.

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

44. Diffie-Hellman Protocol: Let A, B be two users. Note that everything but x and y are known. We will work in the group $\mathbb{Z}_N = \{1, \ldots, m-1\}$.

    (a) A: Pick $x \in \{0, m-1\}$. Set $h_1 = g^x$, and send it to B.

    (b) B: Pick $y \in \{0, m-1\}$. Set $h_2 = g^y$, and send it to A.

    (c) Now, both users can produce shared key $g^{xy} = h_2^y = h_1^x$.

    (d) $c = k * m$, $m = c * k^{-1}$

    If the Decisional Diff & Hellman Assumption holds, then this protocol is secure.

45. Public Key (Asymmetric) Encryption Scheme: Consists of 3 PPT algorithms Gen, Enc, and Dec.

    (a) Gen: Key-generation algorithm that on input 1^n, outputs p_k, s_k.

    (b) Enc: Encryption that on input p_k and message m outputs ciphertext c (Used by the public).

    (c) Dec: Decryption algorithm that on input s_k and ciphertext m, outputs message m or error (Used by the private party/recipient).

    Correctness Property: $Dec_{s_k}(Enc_{p_k}(m)) = m$

46. CPA-Security for Public Schemes: Fix a public-key encryption scheme Π and an adversary A. We will define experiment $PubK - CPA_{A,\Pi}$ as follows.

    (a) Run $Gen(1^n)$ to get keys p_k, s_k.

    (b) Give p_k to A, who outputs (m_0, m_1) of the same length.

    (c) Choose uniform $b \in \{0, 1\}$ and compute the ciphertext $c \in Enc_{p_k}(m_b)$. Then, give c to A.

    (d) A outputs a guess b', and the experiment evaluate to 1 if $b' = b$.

    We say that Π is CPA-secure if for all PPT adversaries A, the following holds.

    $$\Pr[PubK - CPA_{A,\Pi}](n) = 1] \leq \frac{1}{2} + negl(n)$$

    Note that the encryption oracle is redundant since we already give out the public key.

47. CCA-Security for Private Schemes: Similar to for Public Key Encryption, where you get a decryption oracle.

48. El Gamal PKE Scheme (Follows Diffie-Hellman Key Exchange):

    (a) $Gen(1^n)$: Run $G(1^n)$ to obtain G, q, g. Choose a unifrom $x \in \mathbb{Z}_q$. $p_k = (G, q, g, g^x)$, $s_k = x$.

    (b) $Enc_{p_k}(m)$: Choose uniform $y \in \mathbb{Z}_q$. The ciphertext is $(g^y, (g^x)^y * m)$.

    (c) $Dec_{s_k}(c_1, c_2) : c_2 * c_1^{-x} = c_2 * (g^{-xy})$

    El Gamal is CPA-secure under if the DDH assumption is hard for G, similar to Diffie-Hellman key exchange protocol. It is also secure for multiple messages.

    It is not, however, CCA secure. Given (c_1, c_2), we can decrypt $(c_1, \alpha * c_2)$ to get $(c_1, \alpha * c_2) = (g^y, h^y * (\alpha * m))$, meaning that the ouptuts are malleable.

49. Long Message Public Key Encryption:

    (a) Block-by-block: $(m_1, \ldots, m_\ell) \to (Enc_{p_k}(m_1), \ldots, Enc_{p_k}(m_\ell))$

    If the underlying scheme is CPA-secure for short messages, then this is secure for arbitrary length messages. Also, however, inefficient.

    Note: Public Key Encryption is NOT a block cipher! CTR/CBC-Mode makes no sense while ECB mode is secure for Public Key Encryption.

(b) Hybrid Encryption: Use public-key encryption to establish a shared secret key k, then use k to encrypt the message with a private/symmetric key encryption scheme. Lower ciphertext expansion and amortized efficiency of private/symmetric key. encryption

Let $\Pi = (Gen, Enc, Dec)$ be a public key scheme and $\Pi' = (Gen', Enc', Dec')$ be private. We define Π_{hy} as follows.

i. Gen_{hy} : Gen

ii. $Enc_{hy}(p_k, m)$:

A. Choose $k \in \{0,1\}^n$

B. $c \leftarrow Enc_{p_k}(k)$

C. $c' \leftarrow Enc'_k(m)$

D. Return c, c'

iii. Dec_{hy} : Simply decrypt c for key k, use it to decrypt c'.

If Π is a CPA-secure public key scheme and Π' is a private key scheme, then Π_{hy} is a CPA-secure public key scheme.

(c) Key Encapsulation Mechanism: For Hybrid Encyrption, it is sufficient to have a Key Encapsulation Mechanism rather that takes a public key and outputs a ciphertext c and a key k.

i. Correctness: k can be recovered from c given s_k

ii. Security: k is indistinguishable from uniform given p_k and c. We can also define CPA/CCA-security for this.

If Π is a CPA-secure KEM and Π' is a CPA-secure Private Key Encryption scheme, then the combination is a CPA-secure public key scheme (It suffices for Π' to be EAV-secure).

50. El-Gamal Based KEM

    (a) $Gen(1^n)$: Run $G(1^n)$ to obtain G, q, g. Choose a uniform $x \in \mathbb{Z}_q$. $p_k = (G, q, g, g^x)$ and $s_k = x$.

    (b) $Ecaps_{p_k}$: Choose a uniform $y \in \mathbb{Z}_q$. The ciphertext is g^y, the key is $k = H(g^{xy})$.

    (c) $Decaps_{s_k}(c)$: Outputs $k = H(c^x)$

    We then combine this with private key encryption for the full scheme. If DDH holds and H is modeled as a random oracle, then the KEM is CPA-secure. This is CCA-Secure if Enc' is CCA-Secure. It is standardized as DHIES/ECIES.

51. Plain RSA Encrypion Scheme:

    (a) $Gen(1^n) : (N, e, d) \leftarrow RSAGen(1^n)$. We let $p_k = (N, e)$ and $s_k = d$.

    (b) $Enc_{p_k}(m) : c = [m^e \mod N]$

    (c) $Dec_{s_k}(m) : m = [c^d \mod N]$

    This scheme is deterministic, and thus not CPA-Secure. It is also not CPA-secure (Consider encoding $\alpha * m$).

52. PKCS #1 v1.5: Fixes plain RSA by use randomized encoding r to attach to messages.

    (a) $Enc_{p_k}(m) : c = [(r||m)^e \mod N]$, where r is uniformly picked at random.

    (b) $Dec_{s_k}(m) : m = [(r||c)^d \mod N]$, then remove the padding r.

    But, clever CPA attacks are possible, and CPA attacks are known if r is too short.

53. PKCS #1 v2.0: Fixes the prior version. Has not been proven insecure, but like the prior, its security is an open question. No proof of security, unless m is very short.

    (a) Optimal Assymetric Encryption Padding applied to message first. This introduces redundancy, so that not every $c \in \mathbb{Z}_N^*$ is a valid ciphertext.

    Is CCA-Secure under the RSA-assumption and if G, H are modeled as random oracles. Widely used today.

54. RSA-Based KEM

    (a) Encaps(m)
        i. Choose uniform $r \in \mathbb{Z}_N^*$
        ii. $c = [r^e \mod N]$
        iii. $k = H(r)$

    (b) Decaps(c)
        i. $r = [c^d \mod N]$
        ii. $k = H(r)$

    Can be proven CCA-Secure under the RSA assumption, if H is a random oracle.

55. Digital Signatures: Provide integrity in public-key settings, analogous to MACs.

    (a) Gen : Takes as input 1^n, outputs p_k, s_k.

    (b) $Sign$: Takes as input s_k,m, outputs signature r.

    (c) $Vrfy$: Takes p_k and m, outputs 1 or 0.

    Correctness Property:

$$Vrfy_{s_k}(m, Sign_{s_k}(m)) = 1$$

56. Existential Unforgeability: Given some A and Signature Scheme Π, we define random experiment $Forge_{A,\Pi}$.

    (a) $p_k, s_k \leftarrow Gen(1^n)$

    (b) A is given p_k, interacts with oracle $Sign_{s_k}()$. Let m be the set of messages sent to this oracle.

    (c) A outputs (m, r).

    (d) A succeeds and the experiment evaluates to 1 if $Vrfy(m, r) = 1$ and $m \notin M$.

    We say that Π is secure for all PPT attackers A if the following holds.

    $$Pr[Forge_{A,\Pi}(n) = 1] \leq negl(n)$$

    Note that we are still vulnerable to replay attacks, which can be addressed just like in MACs.

57. Hash-and-Sign Paradigm: A scheme for arbitrary length messages. Given signature scheme Π and hash $H : \{0,1\}^* \rightarrow \{0,1\}^n$, we can construct a signature scheme Π' for arbitrary length messages as follows.

    (a) Gen' : Gen

    (b) $Sign'_{s_k}(m) : Sign_{s_k}(H(m))$

    (c) $Vrfy'_{p_k}(m, r) : Vrfy_{p_k}(H(m), r)$

    If Π is secure and H is collision resistant, then Π' is secure.

58. Plain RSA Digital Signature

    (a) $Gen(1^n) : p_k = (N, e),\ s_k = d.$

    (b) $Sign(m) : \sigma \leftarrow m^d \mod N$

    (c) $Verify_{p_k}(m, \sigma)$:1 if $m = \sigma^e \mod N$, otherwise 0

    Attack: If (t_1, t_2) are valid signatures on (m_1, m_2), then we can get pair $((m_1 * m_2), (t_1 * t_2))$.