# Cryptography Review Part 2

* Cryptographic mechanisms
  + Data confidentiality: Encryption – for confidentiality and limited data integrity
  + Data integrity: Hash functions, Message Authentication Codes (MACs)
  + Non-repudiation: Digital signatures- for data origin authentication and non-repudiation
  + Authentication: – for entity authentication
  + Access Control- TODO

Public-Key Cryptography (PKC)

### Trap-door One-Way Functions:

* Definition:A function *f* with the following properties:
  + Ease of Computation: *f(x)* is easy to compute for any *x*.
  + “Trap-door Pre-image Resistance”: Given *y = f(x)* it is computationally infeasible to determine *x* unless special information used in the design of *f* is known.
    - When this trap-door *k* is known, there exists a function *g* which is easy to compute such that *x = g(k,y)*

### Definition and ingredients of a public key cryptosystem (PKC), services provided by public key cryptography

* PKC is a solution to the *key establishment problem* (the other is the Diffie-Hellman key agreement protocol)
* Idea of Public-Key Cryptography
  + Every user has *two* keys:
    - Encryption key is public (so everyone can encrypt messages)
    - Decryption key is only known to the receiver
* Finding the decryption key from the encryption key should be computationally infeasible

Timeline

Description automatically generated

* Definition**:** A PKC consists of a plaintext space , a ciphertext space , a public key space , and encryption functions , indexed by public keys , with the following properties:
  + For every public key , there is a private key such that the encryption function has a left inverse , i.e.
  + Encryption, , and Decryption, , are easy to compute when public key, , and private key, , are known.
  + Given public key , encryption , and ciphertext , it is computationally infeasible to find plaintext or private key
  + Note: by the three properties: encryption is a trap-door one-way function with trap door private key

Diagram

Description automatically generated

* Note: In PKC, it is *not* necessary for the key channel to be secure
* One disadvantage of deterministic PKCs is that identical messages always encrypt to the same ciphertext (like in ECB mode)
  + Particularly problematic if message space is small (like a yes/no vote)
  + Probabilistic or randomized encryption is stronger level of security (more on this later)
* Properties of PKC: messages encrypted using PKC contain sufficient information to uniquely determine the plaintext and the key (given enough ciphertext, resources, etc.)
  + The entropy contained in these systems is zero
  + This is the exact opposite of a perfectly secret system like the one-time pad
* Computational Security: Security in a PKC is only dependent on the computational cost of computing the plaintext and/or private key from the ciphertext
* Ingredients of RSA:
  + Select two distinct large primes and (each around )
  + Computes and
  + Selects a random integer (so and
  + Solves the linear congruence

for

* + Keeps ,, secret and makes and public:
    - The public key is
    - The private key is (or see Assign 3 Q2)
* Encryption:
  + Messages for the designer are integers in
  + E.g. divide a bit string into blocks of bit length
  + Interpret each block as an integer with via its binary representation
  + To send encrypted, compute and send where
* Decryption:
  + To decrypt , the designer computes where
* Both encryption and decryption are modular exponentiation (same modulus, different exponents)

Graphical user interface, text, application

Description automatically generated

### RSA, security of “textbook" RSA: equivalence of factoring n, computing phi(n) and finding d, choice of secure parameters

* RSA Problem (extracting e-th roots modulo n):
  + Given and with
* Integer Factorization Problem (IFP):
  + Given an integer , find a non-trivial factor of
    - If an adversary can solve an instance of the IFP, she can solve the RSA problem (by factoring and finding the private key in the same way as the designer)
    - It is unknown if there are ways of solving the RSA problem without factoring (or solving one of the other equivalent problems listed below).
* Total Breaks of RSA

Text, table

Description automatically generated

* + All three approaches are computationally equivalent:
    - If one can be achieved, any of the other two one can be achieved with very little computational overhead
    - So, there are three equally good trapdoors here: and
  + There is no proof that RSA is secure
    - No proof that factoring is hard
    - Not proved that other methods to solve the RSA problem exist which do not rely on factoring (i.e. not known whether breaking RSA is equivalent to factoring *n*)
  + We need to design RSA system such that cannot be factored easily
* Choices for secure parameters:
  + **Requirements for and :**
    - Probable primes with high probability (say ) – use a good probabilistic primality test.
    - Large: at least (so n is 3072 bits)
    - Not too close together; for
    - and must be *strong* primes, i.e. all have a large prime factor
      * E.g. pick a Sophie Germain prime (so is a safe prime) so that is prime or has a large prime factor; same for
        + May also want to choose to have a large prime factor to avoid *cycling* attacks (where a modest number of repeated encryptions “cycle back” to the plaintext)
        + Choosing random may be sufficient
  + **Requirements for :**
    - For efficiency reasons, is often chosen small; a popular choice is (great for binary exponentiation, only two ‘1’ bits).
    - Beware of really small for certain applications!
    - In practice, can use , but only when RSA is used in conjunction with a secure padding mechanism (e.g. OAEP)
  + **Requirements for :**

### Multiplicative attacks on “textbook" RSA (CCA, meet-in-the-middle), basic idea of RSA-OAEP (no details) and why/how it fixes problems of “textbook" RSA

Multiplicative Attacks on RSA

* “textbook” RSA is not secure against multiplicative attacks
* Multiplicative (or homomorphic) properties of RSA:
* i.e. the encryption of a product is the same as the product of the encryption
* This means that a factorization of plaintext implies one of the corresponding ciphertext, which can be exploited in two attacks

Adaptive CCA on RSA

* An attacker wishing the decryption of of some RSA ciphetext proceeds as follows:
  + Generates with
  + Computes (this is the chosen ciphertext; note that )
  + Obtains the corresponding plaintext:
  + Computes , where is the inverse of

Meet-in-the-Middle Attack on RSA (Passive)

* If for some bit length , then with non-negligible probability, is compositie and satifies with ,
  + The probability that a number of 40-64 bits factors into equal-size factors is between 18 and 50 percent
* The adversary builds a list and a list of all their inverses (mod n)
  + She then computed and searches for a match (here is the inverse of )
  + If for some , then and hence
* Requires modular exponentiations (rest is negligible).

Graphical user interface, text, application

Description automatically generated

RSA-OAEP

* Optimal Asymmetric Encryption Padding (OAEP)
  + An invertible transformation from a PKC plaintext space to the domain of a one-way trapdoor function (e.g. a public key encryption map)
  + OAEP augments PKCs to provide plaintext awareness by adding redundancy and transforming the plaintext before encryption.
* Can be proven to be plaintext-aware assuming that the RSA problem (computing e-th roots modulo n) is intractable:
  + Defeats CCAs because only messages with the prescribed redundancy ( appended) are accepted. Probability of a random ciphertext decrypting to an acceptable value is
  + Plaintext is randomized – prevents small message space attacks ( possible encryptions of each message).

### Randomized encryption | you should be able to work with El Gamal key generation, encryption, and decryption procedures, but you need not remember the details. You don't need to know the details of the Goldwasser-Micali PKC.

* Probabilistic or randomized encryption utilizes randomness to attain a provable, stronger level of security
* This results in every message having many possible encryptions
  + Leads to semantic security

EL Gamal PKC

* Security based on discrete logarithmic problem (DLP) – alternative to RSA which is based on integer factorization problem (IFP)
  + DLP: Given and with , find
* Setup: the designer produces her public and private keys as follows:
  + Selects a large prime and a primitive root of
  + Generates a random integer with and computes where
  + Public key:
  + Private key: {x}
* Note: multiple users may use the same and , but everyone should have their own pair

El Gamal Encryption

* To send encrypted where (so ),
  + Select a random
  + Compute and send where

El Gamal Decryption

* To decrypt , the designer computes
* Think of as a “clue” that can be used to remove the “mask” in thus “unmasking” the encrypted message

Graphical user interface, text, application

Description automatically generated

### Facts and security assumptions for El Gamal and Goldwasser-Micali

El Gamal

* Security of this system relies on the presumed difficulty of the DLP, unknown if there are other ways of breaking El Gamal
  + Disadvantages:
    - Message expansion by a factor of 2 (ciphertext twice as long as plaintext)
    - Twice as much computational work for encrypting as RSA:
      * Two exponentiations (and one multiplication), as opposed to one exponentiation only for RSA.
    - A new random number must be generated for each message
  + Advantages:
    - Randomized
    - Different security assumptions
    - Working in other settings (e.g. Elliptic curves)
* PKC can only be semantically secure if it is also polynomially secure
* Semantic Security: A PKC is said to be semantically secure if for all probability distributions over the message space, anything that can be computed by a passive adversary is expected polynomial time about the plaintext given the ciphertext can also be computed in expected polynomial time without the ciphertext.
  + - Semantic security is a weaker version of perfect secrecy
    - An adversary with polynomially-bounded computational resources (as opposed to infinite resources in perfect security) can learn nothing about the plaintext form the cipher text
* Polynomial security (IND-CPA secure): if no passive adversary can in expected polynomial time select two plaintexts and and then correctly distinguish between encryptions of and with probability significantly greater than

Goldwasser-Micali

* Achieves semantic security assuming the intractability of the Quadratic Residuosity Problem (QRP)
  + Private Key: where and are distinct large primes
  + Public Key: where and is the pseudo-square modulo
* Polynomial security since is selected at random:
  + is a random quadratic residue modulo
  + Thus is a random pseudosquare modulo
* The cryptanalyst only sees a sequence of or (quadratic residues and pseudosquares), and as the QRP is hard, she cannot distinguish one from the other
* Major disadvantages:
  + Huge message expansion, by a factor of : a t-but message yields a ciphertext of
  + Costly decryption algorithm (t Legendre symbols)

Formal Notions of security for public key cryptosystems

* To address CPA, we need stronger security notions than semantic/polynomial security:

### Security against passive attacks: polynomial/IND-CPA and semantic security

IND-CPA

* Security levels: indistinguishability under chosen plaintext attacks (CPA) (same as polynomial security)

### Security against active attacks: IND-CCA2 security, non-malleability, plaintext awareness and how they relate to each other logically

IND-CCA1

* Definition: A PKC is IND-CCA/IND-CCA1 secure if it satisfies indistinguishability under CPAs; in other words, no (active) adversary with black box access to a decryption oracle (that decrypts arbitrary ciphertexts) can in expected polynomial time select two plaintext messages and with probability significantly greater than
* Same definition as a polynomial security except that access to a decryption oracle is granted. It is an active attack equivalent of sematic security

IND-CCA2

* Definition: A PKC is IND-CCA2 secure if it satisfies indistinguishability under adaptive CCA, i.e. an attacker may use the decryption oracle adaptively (of course as always, she may not submit the encryption given to her to distinguish from )
* Adaptive CCA strategy is permitted:
* Security levels:
  + IND-CCA2: indistinguishability under adaptive CCA
  + IND-CCA: indistinguishability under (non-adaptive) CCA
  + IND-CPA
  + Note that

Non-malleability

* Definition: A PKC is non-malleable if, given a ciphertext corresponding to some message , it is computationally infeasible to generate a different ciphertext whose decryption is related to in some known manner, i.e. for some arbitrary but known (efficiently invertible) function .
* Non- malleability provides data integrity of ciphertexts without any source identification (public-key analogue of “encrypt-then-MAC”)
* We have
  + It is known that and

Plaintext Awareness

* Very strong notion of security
* Definition: a PKC is plaintext-aware if it is computationally infeasible for an adversary to create a valid ciphertext without being aware of the corresponding plaintext
* A plaintext aware PKC resists adaptive CCAs because any adaptive modification of a target ciphertext will with high probability not be “valid”
* Plaintext awareness Indistinguishability
* Plaintext awareness Non-malleability

Signatures

### Definition, of properties of, and services provided by digital signatures

Digital signatures

* Definition: Data origin authentical is achieved by means of a **signature** i.e. a means by which the recipient of a message can authenticate the identity of the sender
* Properties: A means for data origin authentication that should have two properties:
  + Only the sender can produce their signature
  + Anyone should be easily able to verify the validity of the signature.
* Services:
  + Non- repudiations: anyone can resolve a dispute where a receiver claims the sender signed the message, whereas the singer claims they didn’t
  + Only sender can generate a signature, but anyone can verify it (not like MAC)
  + Prevent replay attacks using time stamp/sequence numbers in signature

### Signatures from PKCs, impersonation attack on “textbook" PKC-based signatures and how to prevent this attack by using a cryptographic hash function

Signature Capability

* Definition: A PKC is signature capable if and for all
  + In signature capable PKC, decryptions are right and left inverses (i.e. actual inverses) of encryptions
* RSA has signature capability. El Gamal and Goldwasser-Micali do not.

Existential forgery

* Definition: a signature scheme is susceptible to existential forgery if an adversary can forge a valid signature of another entity for at least one message
* Goal of the attacker:
  + Total break: recover private key
  + Universal forgery: generate a signature for any message
  + Selective forgery: generate a signature for some message of choice
  + Existential forgery: generate a signature for at least one message

Existential forgery on PKC- Generated Signatures

* Generate a signature to a message using a signature- capable PKC as described above. Eve can create a forged signature from Alice as follows:
  + Select random
  + Compute
  + Sends to Bob
* Bob computes and accepts the “signature” to “message”
* Usually foiled by language redundancy, but may be a problem if is random (e.g. A cryptographic key)

Preventing Existential Forgery Attack

* Solution to the above:
  + Alice sends where is a public pre-image resistant hash function on .
  + Bob computes and , and accepts the signature iff they match.
* Foils the attack:
  + If Eve generates random , then she would have to find such that (i.e. a pre-image under ), and send to Bob
  + Bob then computes and compares with
  + Not computationally feasible if is pre-image resistant

Existential Forgery if is not Collision Resistant

* Suppose Alice uses a pre-image resistant hash function as described above to sign her messages. If is not collision resistant, Eve can forge a signature as follows:
  + Find with and (a collision)
  + If is the signature to , then is also the signature to , as
* Note that if Eve intercepts , then she could also find a weak collision with

### El Gamal signature scheme | you should be able to work with El Gamal key generation, encryption and decryption procedures, but you need not remember the details

El Gamal Signature Scheme

* Variation of El Gamal PKC, Alice produces her public and private keys as follows:
  + Selects a large prime and a primitive root of
  + Randomly selects such that and computes
* Public key:
* Private key:
* Alice also fixes a public cryptographic hash function

Signing and Verifying

* Alice signs a message as follows:
  + Selects a random integer
  + Computes ,
  + Solves for
  + Alice’s signature is the pair
* Bob verifies Alice’s signature asfollows:
  + Obtains Alice’s authentic public key
  + Computes and
  + Accepts the signature iff and

### Security assumptions for El Gamal

* El Gamal hashes both and which is GMR-secure in the ROM assuming that takes on randoom values and computing discrete logarithms modulo is hard
  + Random Oracle Model (ROM: security proofs relying on random oracle functions)
  + Random Oracle: mathematical functions mapping every possible query (input) to a random response form its output domain (output)
  + DLP reduces to forgery, i.e. an algorithm for producing existential forgeries can be used to solve the DLP
* Public parameter must be chosen verifiably at random (publish PRNG, see, and algo used) in order to ensure that is a primitive root of (applies to Diffie-Hellman and El Gamal cryptosystem as well)

### Notions of universal, selective, and existential forgery

* Total break: recover private key
* Universal forgery: generate a signature for any message
* Selective forgery: generate a signature for some message of choice
* Existential forgery: generate a signature for at least one message

### Definition of GMR-security

* GMR stands for Goldwasser-Micali-Rivest
* Definition: A signature scheme is said to be GMR-secure if it is existentially unforgeable by a computationally bounded adversary who can mount an adaptive chosen message attack.
* In other words, an adversary who can obtain signature of any message of her own choosing from the legitimate signer is unable to produce a valid signature of any new message (for which she has not already requested and obtained a signature) in polynomial time.

**Pseudorandom number/bit generation**

### Definition of a PRNG/PRBG, idea of a seed

Pseudorandom Number/Bit Generator

* Definition: An algorithmic technique to create sequences of statistically random number/bits, initialized with a random seed

### Definition of cryptographically secure PRGB, next-bit test

Cryptographically secure PRGB (CSPRBG)

* Must pass the **next-bit test**: there is no polynomial time algorithm that, on input of the first bits of an output sequence, can predict the -st bit with probability significantly greater than ½
* For all practical uses, CSPRBG is unpredictable
* Remark: A PRBG is cryptographically secure iff it passes the **previous bit test**: there is no polynomial time algorithm that, on input of buts of an output sequence can predict the preceding bit with probability significantly greater than ½.

### Entropy and proper use of PRNGs

* Security of a PRNG is determined by the entropy of its seed (which is the bit length if the seed is random). So, the seed must have sufficient **entropy** to make the PRNG unpredictable.
* Number of bits of entropy must correspond to the overall bit security of the system.
* Ex: 3072-bit RSA provides 128-bit security, so the seed material for the PRNG must have at least 128 bits of entropy

Table

Description automatically generated

* Key length for block cipher providing equivalent level of difficulty to break (first two security levels, 80 and 112, are now considered insufficient)

Key management

### Basic idea of symmetric key distribution via key distribution centres

Symmetric key

* Requires both parties to share a common secret key

Key Distribution Centres (KDC)

* Idea:
  + Each user holds a shared symmetric **master key** with the KDC
  + Master keys are used for distributing one-time **session keys**:
    - KDC generates a session key shared between and
    - KDC encrypts with master key it shares with and sends encryption to
    - KDC encrypts with master key it shares with and sends encryption to
  + A and B communicate using session key for encryption and destroy at the end of their session
* Advantages:
  + Far fewer long-term keys than if each pair of entities holds a shared long-term key
  + Compromise of a session key does not affect master key nor other sessions
* Issues:
  + Session key lifetimes should be limited for greater security
  + All keys and entities (users and KDCs) must be authenticated

### Basic ideas of public key distribution by means of public key infrastructures; notions of certificates, certification authorities and their purpose

Public-Key Infrastructures (PKI):

* Definition: A set of techniques and procedures supporting authenticated key management for PKC. Specifically, a PKI supports:
  + Initialization of system users
  + Generation, distribution/authentication, and installation of public and private keys
  + Controlling the use of keys (e.g. life cycles of session keys, public and private keys)
  + Update, revocation, and destruction of keys (e.g. managing compromise of private keys)
  + Storage, backup/recovery, and archival of keys (e.g. maintaining an audit trail)

Public-Key Certificates

* Definition: A data structure consisting of a **data part** (containing at least the user ID and the corresponding public key) and a **signature part** consisting of the digital signature of a *certification authority* over the data part
* A certificate should also include information such as:
  + A timestamp indicating the currency of the certificate (to facilitate key changing and revocation)
  + Additional information about the key (Key generation algorithm, intended use)
  + Key status (for revocation)
  + Signature verification information (certification authority’s name, signature algorithm used)

Certification Authorities

* Definition: A trusted third party whose signature on a certificate vouches for the authenticity of the public key bound to the subject entity
* Idea: CA issues public key certificates that may be verified off-line. Users may exchange authentic public keys without having to contact the CA.

Graphical user interface, text, application

Description automatically generated

Entity authentication

### Services provided by entity authentication protocols

Authentication Protocols and Nonces

* Definition: A sequence of one or more information exchanges used to convince parties of each others’ identity
* Authentication may be one-way or mutual. Key issues:
  + Confidentiality (e.g. to protect session keys)
  + Timeliness (freshness) – to prevent replay attacks where a signed message is copied and later resent
    - Ensured via time stamps or nonces

Nonce

* Definition: A number or bit string that is used only once (usually in a particular message or protocol)

Graphical user interface, text

Description automatically generated

### Station-to-Station protocol | you need not remember details, but you should know its purpose, what cryptographic services it provides, and that it is an authenticated version of Diffie-Hellman

Services provided by STS AKA authenticated Diffie-Hellman key agreement

* Mutual entity authentication (via signed user IDs)
* Mutual authenticated key agreement – each party contributes randomness to , each partysignsthe key agreement material
* Mutual key confirmation- both parties encrypt and decrypt with
* Perfect forward secrecy - compromise of one session key or even one of the private keys as each session key is generated from one-time secrets
* Note: and also playing the role of nonces to assure freshness

Cryptographic application

### SSH - you need not remember details, but you should know what it is, its purpose, what cryptographic services it provides, and the basic three-layer structure

Secure Shell (SSH)

* Is a PKC-based access control system for remote login and file transfer that consists of 3 components:
  + SSH Transport Layer Protocol (TLP)
    - Algo negotiation
    - Unilateral authentication (server to client) – client downloads server’s public key
    - Establishment of shared session key for secure communication
  + SSH User Authentication Protocol
    - Unilateral authentication (client to server) protected by shared session key
  + SSH Connection Protocol
    - Interactive applications protect by shared session key

Management and Validation of Server’s Public Keys

* Two approaches: PKI or local database
  + Superior solution: public-key certificates
    - Problem: PKI not widely deployed
  + Current solution: Each client maintains a local database containing associations between servers and public keys
  + Suggested methods to ensure authenticity of store public keys:
    - Carry authenticated copy on removable storage media (USB key or token)
    - Obtain public key over an insecure channel, verify over phone (read out hash of obtained public key - unfortunately, this is generally not done)

Number theory

* Basic proof techniques
* and ­­, Euler phi function
* Congruences and integer modular arithmetic
* Modular arithmetic on polynomials with binary coefficients
* Base 2 logarithms Euler's and Fermat's Theorems, primitive roots
* Euclidean algorithm and extended Euclidean algorithm, modular inverses, solving linear Diophantine equations and linear congruences
* Binary exponentiation
* Quadratic residues and non-residues, Euler's criterion

### Legendre and Jacobi symbols and their properties, efficient computation of Legendre and Jacobi symbols, the Quadratic Residuosity Problem

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Quadratic Residuosity Problem

* Definition: Given an odd composite integer and any with , determine whether .