* Keys are not based on substitution/transposition like with symmetrical keys. They are based on mathematically hard problems.
* The public and private keys are related mathematically.
* Example of public key ciphers: RSA, DSS (Digital Signature Standard), DH (Diffe-Hellman), etc.

Using PKC to achieve confidentiality

* Encrypt using the receiver’s public key.
* Receiver can then decrypt using their private key. Which they only have access to.
* Small sequences encrypted for confidentiality. Usually symmetrical keys.

Using PKC to achieve authenticity

* Encrypt M with the sender A’s private key
* The receiver B has access to A’s public key
* The receiver can decrypt M with the public key. If successful, then this confirms to B that M has been sent by A.
* Signature is usually signed on the hash of M. Why? Because the hash is usually smaller than M (making encryption and decryption computationally efficient). Signing the hash achieves both integrity of M and authentication.

Features of PKC

* Based on the idea of a trapdoor function AKA mathematically hard problems (factoring large composites of primes, discrete logarithms etc.)
* It is easy to generate public and private keys
* Easy to encrypt and decrypt if the correct keys are known.
* It is difficult to compute private key from public key.
* Hard to recover plaintext from cipher text without the correct key.

Modulo Operator

* Interested in the remainder (AKA residue) left ‘b’ when ‘a’ is divided by ‘n’
* A = b (mod n) <=> a = k\*n + b
* 0 <= b <= n-1
* Add modular arithmetic properties image.
* Multiplicative inverse in modular arithmetic a\*x = 1 (mod n). This means that need to find x in [0, n-1] such that when you divide it by n then you get a remainder of 1.
* Example: 3 \* 4 mod 11 = 12 mod 11 = 1, 3 and 4 are each other’s multiplicative inverses (mod 11)
* Iff a and n are **relatively prime**, gcd(a, n) = 1, then a has a **unique inverse - x** (mod n) given that a has an inverse.
* If n is prime then, all a in [0, n-1] is relatively prime to a -> all ‘a’ have a unique inverse.

RSA Algorithm

* Easy to understand and implement
* Algorithm consists of 2 numbers. The modulus n and the public exponent e.
* **Block cipher**
* The modulus is a product of 2 large prime numbers p and q. p and q are **secret**
* 3 steps to the algorithm: Key generation, Encryption and Decryption.
* Plaintext and ciphertext are numbers between 0 and n-1

Algorithm – Key generation

* Select 2 large primes p and q.
* Calculate n = p\*q and phi(n) = (p-1)\*(q-1)
* n is public and calculated
* Select e (1 < e < phi(n)) relatively prime to phi(n)
* e is public and chosen
* Find the inverse d of e. (d\*e = 1 mod phi(n))
* Public key = {e, n}
* Private key = {d, n}
* d is private

Encryption

* C = M^e mod n

Decryption

* M = C^d mod n

Why RSA works

To be filled

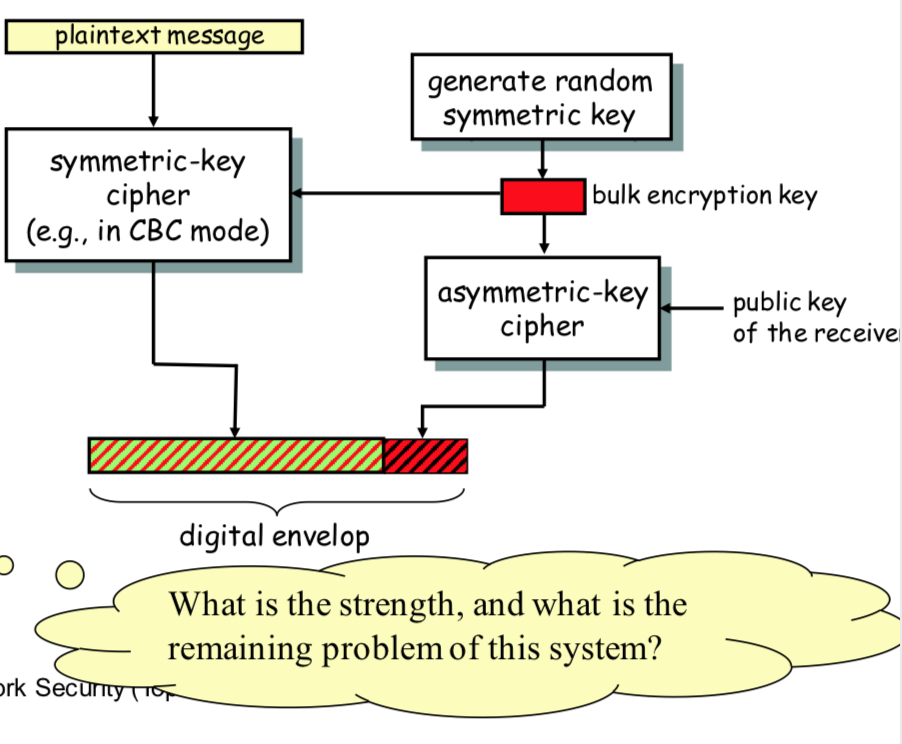
Standard for RSA

* Known as PKCS#1. Defines in the use of RSA algorithm.
* Defines Key Generation, encryption, decryption, digital signatures, verification, public key format, padding and several other issues with RSA.

RSA facts

* Difficult to find d given {e, n}
* If n which is a large prime can be factored (find p and q) then d can be found. But doing this is computationally difficult.

Hybrid Cryptosystems

* Used to solve the problems with both Public key ciphers and symmetric key ciphers.
* Public key cipher are much slower than symmetric key cipher
* Symmetric ciphers have key management problems. And cannot prove non-repudiation service without the involvement of a trusted third party.
* We use public key cipher for symmetric key transportation and digital signature generation. And symmetric keys are used for bulk encryption.