

41900 – Fundamentals of Security

Asymmetric-Encryption and RSA

Ashish Nanda

Ashish.Nanda@uts.edu.au

Asymmetric key cryptography

- Symmetric and asymmetric-key cryptography will exist in parallel and continue to serve the community.
- They are complements of each other; the advantages of one can compensate for the disadvantages of the other.
- Asymmetric key cryptography uses two separate keys: one private and one public.



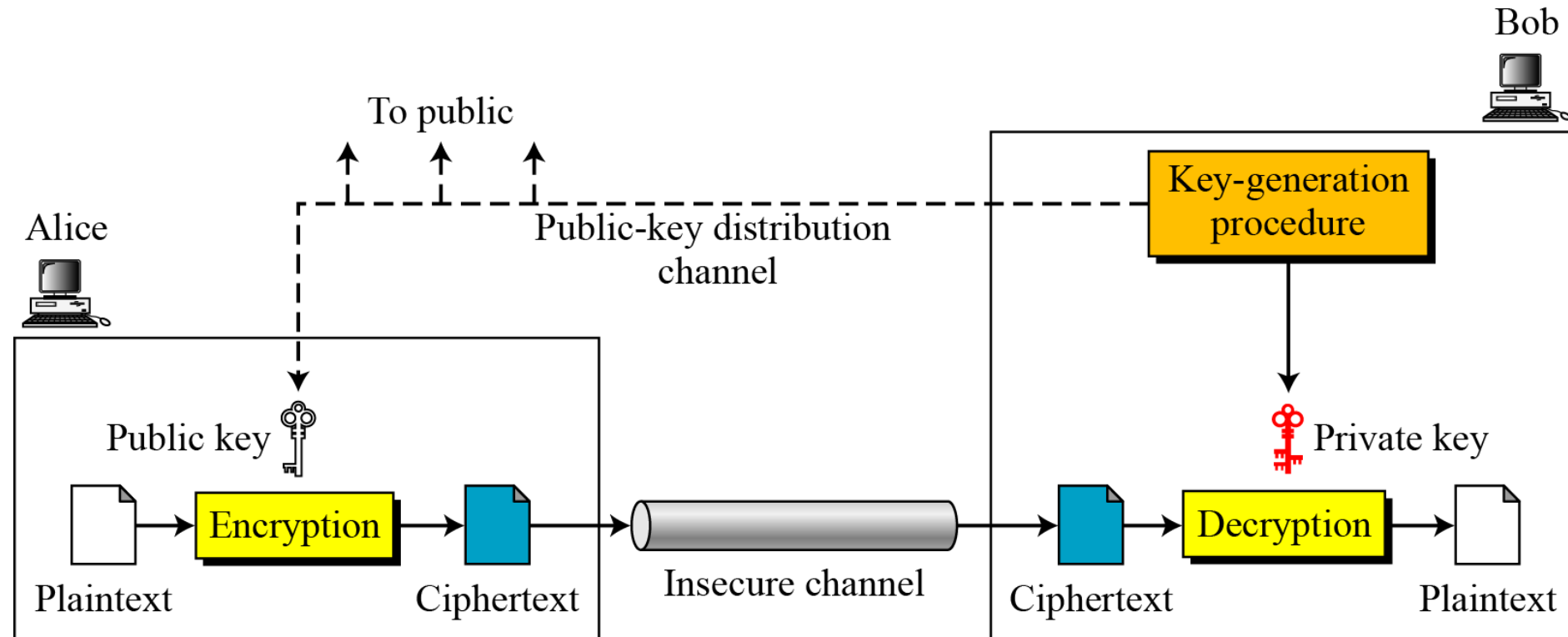
Cipher = f (K_{public} , Plaintext)

Plaintext = g(K_{private} , Cipher)

Public Key Cryptography

- Public key cryptography is a fundamental and widely used technology around the world. It is the approach which is employed by many cryptographic algorithms and cryptosystems.
- Each user has a pair of cryptographic keys: a public key and a private key. The private key is kept secret, whilst the public key may be widely distributed.
- Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key. The keys are related mathematically, but the private key cannot be feasibly derived from the public key.

Asymmetric Cipher



- Also known as public key
- More computationally intensive

Asymmetric cipher

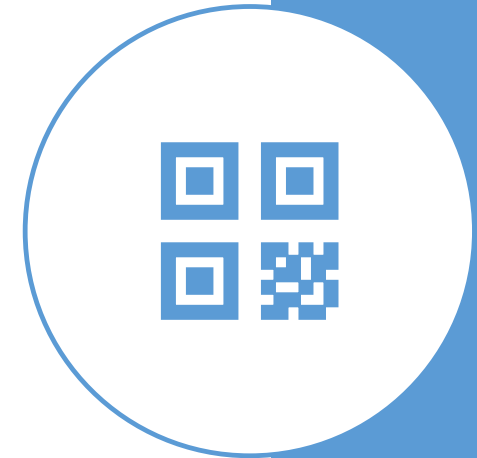
Two most popular algorithms are:

RSA

- Developed by Ron Rivest, Adi Shamir, Len Adelman
- Both public and private key are interchangeable
- Variable Key Size (512, 1024, or 2048 bits)
- Most popular public key algorithm

ElGamal

- Developed by Taher ElGamal
- Variable key size (512 or 1024 bits)
- Less common than RSA, used in protocols like PGP



Why Asymmetric key?

The reason public keys are used is to establish secure communication when there is no way to exchange a key beforehand.

Confidential/authenticated channels for free?

Must ensure that the public key belongs to the correct party (binding of identity to key). The public key directory may be corrupted:

Solution: Use a Public Key Infrastructure to certify your keys (PKI)

Anybody may encrypt messages that only destination device may read, since he knows the private key (K_{private})





Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

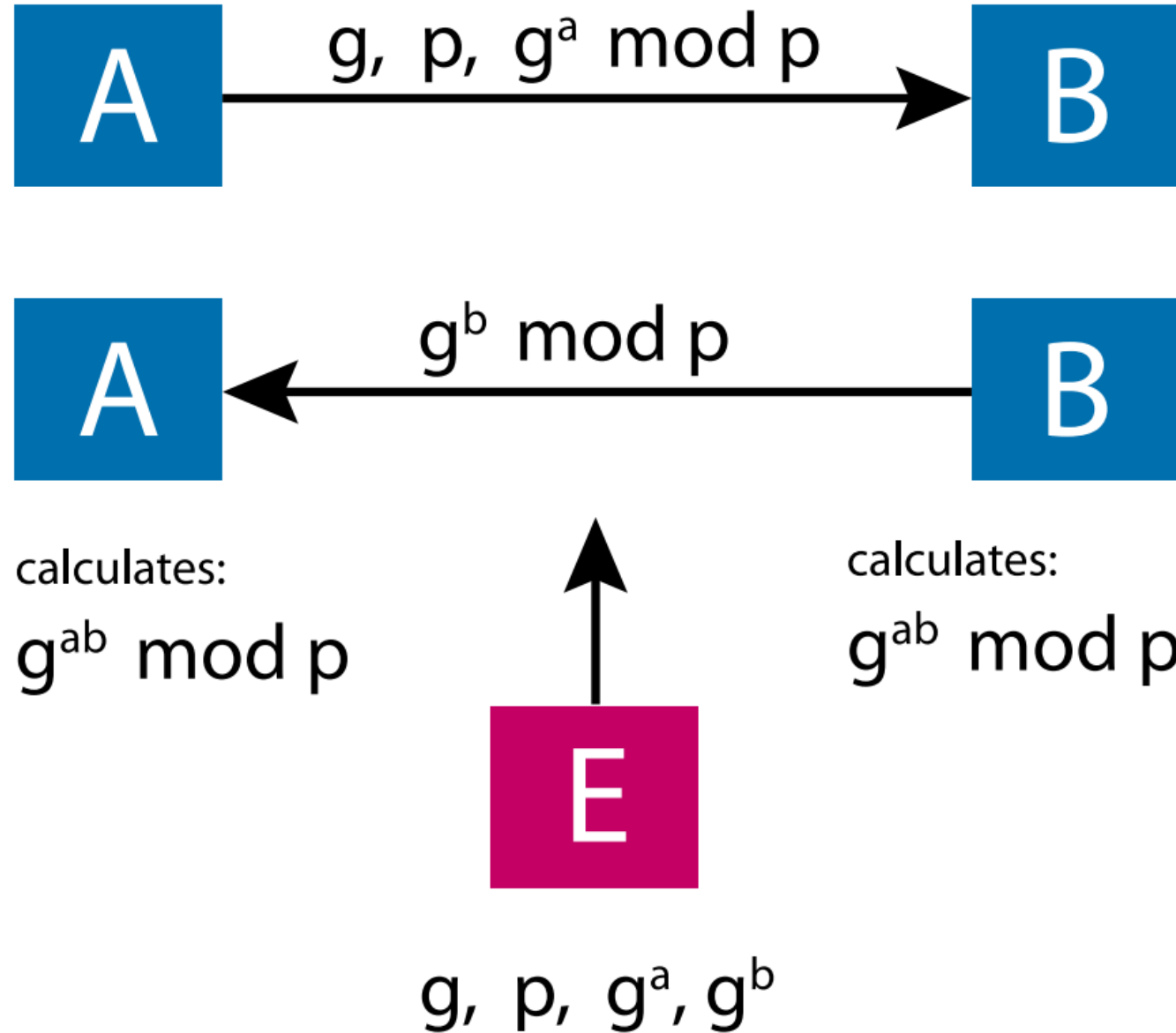
Problem: Alice and Bob, talking on an insecure channel, wish to establish a shared key. Eavesdroppers are present and can read all communication between Alice and Bob (but cannot interfere otherwise). Is it possible for Alice and Bob to establish a shared secret?

Solution: Diffie-Hellman key exchange.

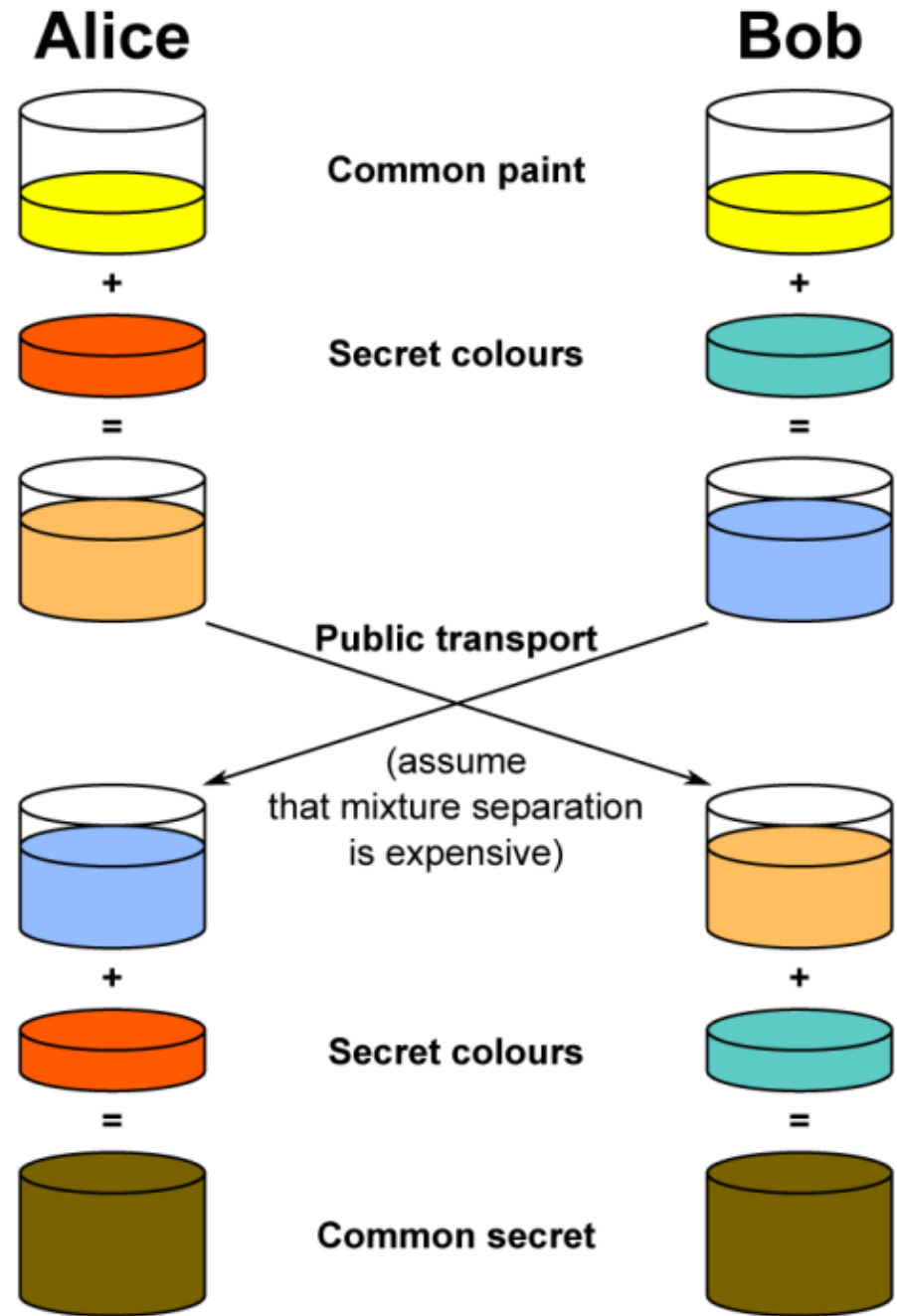
1. Alice and Bob agree on a large prime p , and a generator g of \mathbb{Z}_p^\times
2. Alice picks a random number a ($1 < a < p$) and sends g^a
3. Bob picks a random number b ($1 < b < p$) and sends g^b
4. Alice and Bob both compute $g^{ab} = (g^a)^b = (g^b)^a$
5. The shared secret is g^{ab}



Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange: Paint Analogy



Computing in \mathbb{Z}_p^\times

Computing over \mathbb{Z}_p^\times (where p is prime)

Things that are easy:

- Generate a random number
- Multiply two numbers
- Take powers: $g^r \bmod p$
- Find the inverse of an element
- Solve a linear system
- Solve polynomial equation of degree d

Things that are hard:

- The Discrete Log Problem (**DLP**)
- The Diffie-Hellman Problem (**DHP**)

Hardness of DH

Fix a prime p , and let g be a generator of \mathbb{Z}_p^\times

The discrete log problem (**DLP**) is:

Given $x \in \mathbb{Z}_p^\times$, find $r \in \mathbb{Z}$ such that $x = g^r \pmod{p}$



The Diffie-Hellman problem (**DHP**) is:

Given $g^a, g^b \in \mathbb{Z}_p^\times$, find g^{ab}

Attacks on the Discrete Log Problem (DLP)

Obvious attack: exhaustive search

Linear in the scale of p , since \mathbb{Z}_p^* has $p - 1$ elements.

If p has b bits, $p \approx 2^b$. So $O(2^b)$, where b is the key length.

Various methods are more efficient and can calculate in $O(2^{b/2})$:

- Baby-step giant-step (square root) algorithm
A time-memory trade-off of the exhaustive search method
- Pollard's rho model
- Pohlig-Hellman algorithm

For large values of n , these methods are **not currently practical**. There does exist an efficient quantum algorithm for solving DLP however.



Selecting g and p for Diffie-Hellman

You can select them yourself, but using RFC standards **RFC 3526** is preferred.

A set of Modular Exponential (MODP) groups are defined in RFCs.

This means that instead of exchanging g and p each time, you simply state: **RFC3526 1536-bit**, and both parties know the parameters to use.

- These avoid known attacks against certain classes of primes.

If g and p are chosen well, then the security is in the random choice of a and b .

The Strength of Diffie-Hellman

The strength of Diffie-Hellman is based upon two issues:

- Given p, g, g^a - It is difficult to calculate a
(the discrete logarithm problem)
- Given p, g, g^a, g^b - it is difficult to calculate g^{ab}
(the Diffie-Hellman problem)

We know that **DL \rightarrow DH** but it is not known that **DH \rightarrow DL**

The Strength of Diffie-Hellman

Essentially the security is based on number tricks:

- The strength is proportional to the difficulty of factoring numbers the same size as **p**.
- The generator (**g**) can be small.
- Do not use the secret g^{ab} as a key!



Pro-tip: It is better to either hash it or use it as a seed for a PRNG - not all bits of the secret have a flat distribution!



RSA



RSA: Operation

Key generation is done by a single party, Alice:

- Generate two large primes p , q , and let $n = pq$
- Select e , which is coprime to $\phi(n) = (p - 1)(q - 1)$
- Find d such that $ed = 1 \pmod{\phi(n)}$ using the Euclidean algorithm.
- The public key is (n, e) and the private key is (n, d)
- p , q , $\phi(n)$ can now be thrown away.

If Bob now wants to send the message $m \in \mathbb{Z}_n^\times$ to Alice:

- Bob computes $m^e \pmod{n}$ and sends the result to Alice.
- Alice receives m^e and computes $(m^e)^d \equiv m^{ed} \equiv m \pmod{n}$.

RSA: Example

Alice generates a new key:

- Generate two primes $p = 11$, $q = 17$, and let $n = pq = 187$
- Select $e = 7$, which is coprime to $\phi(n) = (p - 1)(q - 1) = (11 - 1)(17 - 1) = 160$
- Let $d = 23$ such that $ed = 1 \pmod{\phi(n)} = 161 \pmod{160} = 1$
- Public key: $(n, e) = (187, 7)$ Private key: $(n, d) = (187, 23)$

Suppose Bob wants to send $m = 11$ to Alice

- Bob computes $m^e \pmod{n} = 11^7 \pmod{187} = 88$. Send **88** to Alice
- Alice receives **88**, and computes $88^d = 88^{23} \pmod{187} = 11$. She has recovered Bob's message!

Security of RSA: Calculating in Z_n^\times

Computing over Z_n^\times , where $n = pq$ for primes p and q

Things that are easy:

- Generate a random number
- Multiply numbers
- Take powers: $g^r \bmod n$
- Find the inverse of an element
- Solve a linear system

Things that are hard:

- Finding the prime factors of n
- Computing the e^{th} root (as hard as factoring n)
- Solving polynomial equation of degree d

Core security of RSA

The RSA problem

Recovering the value **m** from **c**, where **c = m^e mod n**

(i.e. Taking the **eth** root of **m mod n**)

This is hard: the most efficient means so far known is to factor **n** into **p** and **q**

Integer Factorisation

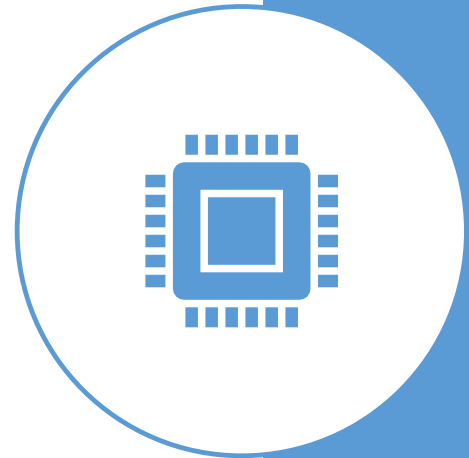
Given a composite number **n**, decompose it into its prime factors

$$P_1, P_2, \dots, P_n$$

No polynomial time algorithm is yet known.

Not proven to be “hard”

- Like DLP and DHP from DH key exchange, we assume the RSA and integer factorisation problems are computationally “hard” but we have no proofs.
- Quantum algorithms already exist that would easily break RSA.



Factoring Attack on RSA

The RSA problem: recover m from $m^e \bmod n$ knowing only n and e .

Suppose n can be factored into p and q :

- Then $\phi(n) = (p - 1)(q - 1)$ can be found.
- Knowing $\phi(n)$, compute the decryption exponent: $d = e^{-1} \pmod{\phi(n)}$.
- This means we now own the private key, and can decrypt anything.

When performing key generation, it is important that the primes p and q are selected to make factoring $n = pq$ very difficult

e.g. by picking a p and q that are roughly equal size



Size of Modulus in RSA

Powerful attacks on RSA include using a quadratic sieve and number field sieve factoring algorithms to factor the modulus $n = pq$.

1977: A 426 bit RSA puzzle was published by Martin Gardner's Mathematical Games in Scientific American.

- Ron Rivest said RSA-125 would take “40 quadrillion years”.

1993: It was cracked by a team using 1600 computers over 6 months: “the magic words are squeamish ossifrage”

1999: a team led by de Riele factored a 512 bit number

Size of Modulus in RSA

2001: Dan Bernstein wrote a paper proposing a circuit-based machine with active processing units (the same density as RAM) that could factor keys roughly 3 times as long with the same computational cost - so is 1536 bits insecure??

- The premise is that algorithms exist where if you increase the number of processors by n , you decrease the running time by a factor greater than n .
- Exploits massive parallelism of small circuit level processing units

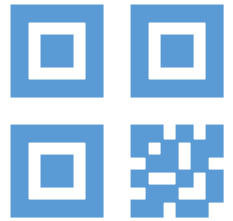
2009: RSA-768, 232 digits (768 bits), is factored over a span of 2 years.

Size of Modulus in RSA

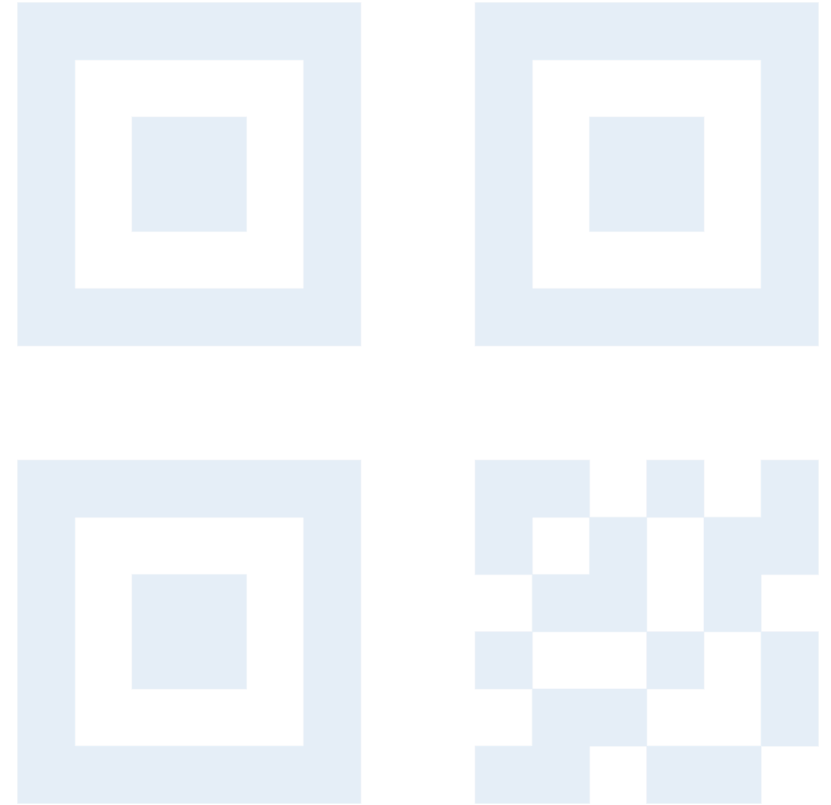
2012: A 1061 bit special number ($2^{1039} - 1$) was factored over 8 months (a special case), using a “special number field sieve”.

It is recommended today that **4096** bit keys be used (or at least **2048** bit) and **p** and **q** should be about the same bit length (but not too close to each other).

Advances in factoring are leaps and bounds over advances in brute force of classical ciphers.



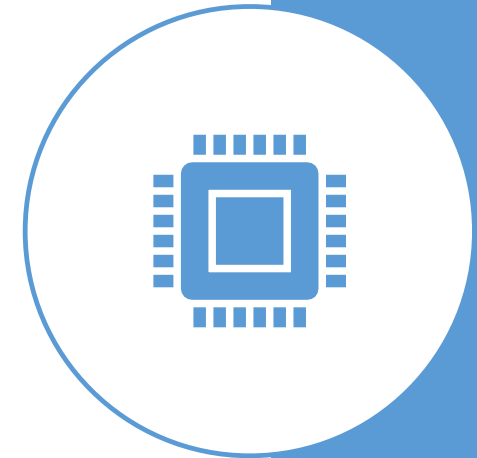
Symmetric vs. Asymmetric



Summary: Symmetric Crypto

Advantages of symmetric-key crypto:

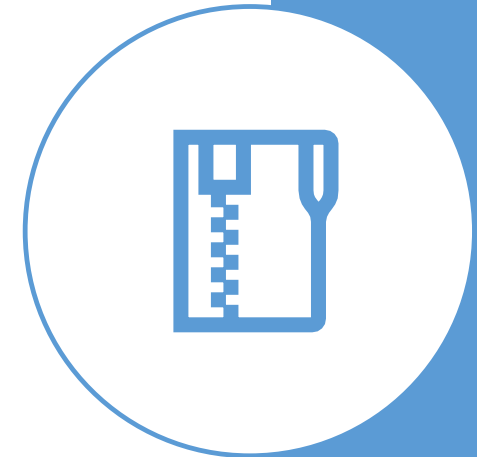
- Can be designed to have high throughput rates
- Keys are relatively short (128, ..., 256 bits)
- Can be used as primitives to create other constructs such as pseudo random number generators (PRNGs).
- Can be used to construct stronger ciphers
 - e.g. simple substitutions and permutations can be used to create stronger ciphers
- All known attacks involve “exhaustive” key search.



Summary: Symmetric Crypto

Disadvantages of symmetric-key crypto:

- In a two party network, the key must remain secret at both ends
- Sound practice dictates the key needs to be changed frequently (e.g. each session)
- In a large network, $\binom{n}{2}$ keys are required, which creates a massive problem for key management



Summary: Asymmetric Crypto

Advantages of asymmetric-key crypto:

- Only the private key needs to remain secret
- The administration of keys on a network requires the presence of only one functionally trusted (honest and fair) TTP.
- Depending on the mode of usage, the public and private key pairs may be used for long periods of time (upper bound: Moore's Law)
- In large networks, n keys are required instead of $\binom{n}{2}$



Summary: Asymmetric Crypto

Disadvantages of asymmetric-key crypto:

- Throughput rates are typically very slow (for all known algorithms)
- Key sizes are typically much larger (1024, ..., 4096 bits)
- Security is based upon the presumed difficulty of a small set of number-theoretic problems; all known are subject to short-cut attacks (e.g. knowing the prime factorisation of n)
- Public key crypto does not have as much of an extensive history in the public world, compared to symmetric.



Combining Cryptosystems

Symmetric and asymmetric crypto are complementary.

Asymmetric crypto can be used to establish a key for subsequent faster symmetric crypto (e.g. a session key)

Alice and Bob can take advantage of the long term benefits of public key crypto by publishing their public keys to a directory.

Asymmetric crypto is good for key management and signatures.

Symmetric crypto is good for encryption and some data integrity applications.

Symmetric Crypto: Key Length

Security of a symmetric cipher is based on:

- Strength of the algorithm
- Length of the key

Assuming the strength of the algorithm is perfect (impossible in practice) then brute force is the best attack.

Cost (USD)	40 bits	56 bits	64 bits	128 bits
\$100k	0.06 sec	1.1 hrs	11.5 days	10^{18} years
\$1M	6.25 ms	6.5 mins	1.2 days	10^{17} years
\$100M	0.06 ms	3.75 mins	17 mins	10^{15} years
\$1B	6.25 μ s	0.4 sec	1.9 mins	10^{14} years

Interesting ways to break symmetric ciphers

Virus / Worm:

- What if a bot net forced a cipher?
- Melissa infected $\approx 800k$ computers
- If we can cracking DES @ 280k keys/s (P4 @ 2.8 Ghz)
- Melissa-DES could brute force the key in 30 hours
- In 2000, a worm did just that.
- Newer botnets have many more computers.

Chinese Lottery

- Say a 1M key/s chip was built into every radio and TV in China
- Each chip is designed to brute force a key sent over the air
- If 10% of the people in China have a radio or TV: the 56 bit DES key space can be exhausted in 12 minutes

Asymmetric Crypto Key Length

Security of all known public key algorithms is based on the presumed difficulty of a small set of number-theoretic problems.

All are subject to short cut attacks. Tomorrow we might find a way to factor easily.

e.g. Quantum? DNA?

1977: Ron Rivest (RSA) said factoring 125 digit number would take 4×10^{16} years

2003: 576 bit (177 digit) number factored

2012: 1061 bit (320 digit) number factored

Designs are being drawn out for optical / quantum sieving machines that could lead to massive optimisations on these numbers in the near future.

Protection	Symmetric	Asymmetric	Hash
Attacks in real time by individuals	32	-	-
Short term protection against small organisations	64	816	128
Short term protection against medium organisations	72	1008	144
Very short term protection against agencies, long term protection against small organisations (2016 – 2017)	80	1,248	160
Legacy standard level (2016 – 2020)	96	1,776	192
Medium-term protection (2016 – 2030)	112	2,432	224
Long-term protection (2016 – 2040)	128	3,248	256
“Foreseeable future” good protection against quantum computers unless Shor’s algorithm applies	256	15,424	512

How long should an asymmetric key be?