Workshop on Side-channel Analysis: Cryptography Concepts and Background

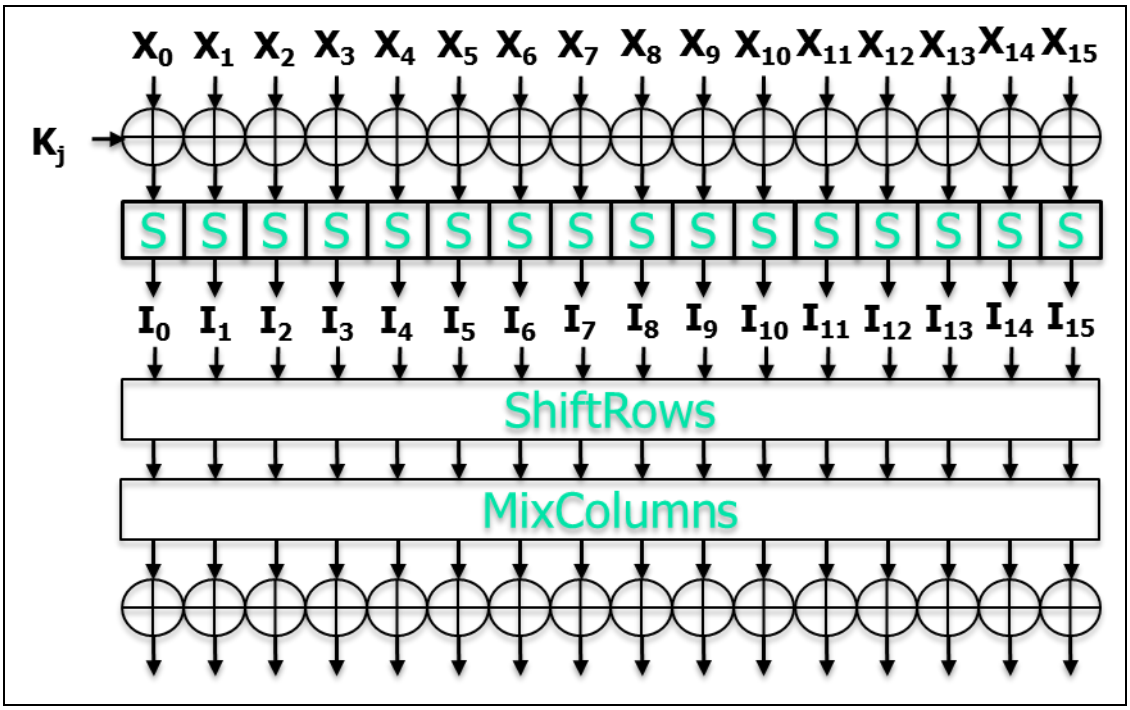
Fundamental Concepts

Symmetric key cryptography

Symmetric encryption algorithms are used to bulk encrypt (scramble) large blocks of data. Modern symmetric algorithms are typically block ciphers. They use a secret key K to encrypt N-bit blocks of data. Modern block ciphers transform the data in a series of rounds. Each round uses a different round key RK, which is typically derived from K. Round transformations typically operate on smaller sub-blocks of the N-bit block. For example, the Advanced Encryption Standard (AES) encrypts 128-bit blocks, but operates on 8-bit blocks during the round transformations.

AES

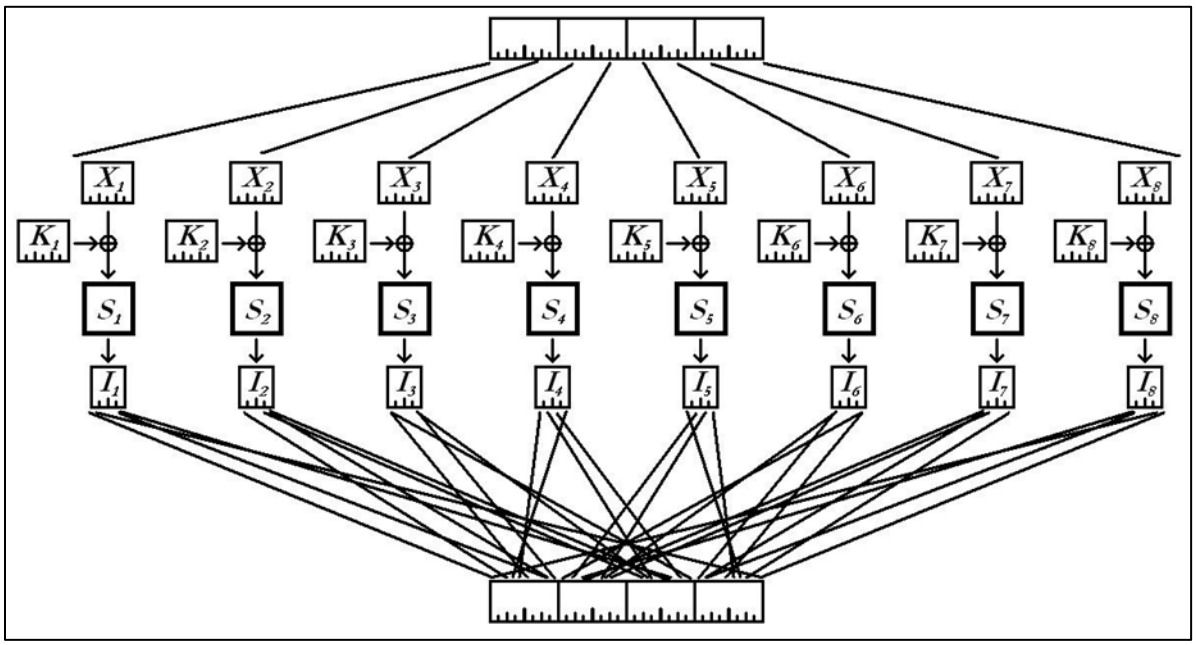
The round structure of AES is shown in the Figure below.



Each byte $\mathbf{X_j}$ is XORed with a byte of key $\mathbf{K_j}$. The resulting intermediates $\mathbf{I_j}$ are transformed by an invertible nonlinear 8-bit lookup table S. The ShiftRows operation permutes the bytes and groups them into four-byte blocks. The MixColumns operation is an invertible linear transformation acting on 4 bytes, and operates on each of the four-byte blocks independently.

DES

Data Encryption Standard (DES) is a Feistel network, in which half of the data (32 bits out of 64) is transformed each round. The main round transformation is show in the figure below.



The 32 bits are expanded to 48 bits, and grouped into eight 6-bit blocks. As with AES, each block $\mathbf{X_j}$ is XORed with a block of key $\mathbf{K_j}$. The results of the XORs are then transformed by 6→4 bit lookup tables $\mathbf{S_j}$. The bits of the resulting intermediates $\mathbf{I_j}$ are then permuted, and the permuted bits are then XORed with the other 32 bits of the data (not shown), to complete the round.

Note that in both algorithms, a small block of data is XORed with a small block of key, and the result is passed through a nonlinear transformation. This construction is often seen in block ciphers, and is useful when performing side-channel analysis.

Parameters for two common algorithms, AES, and DES, are shown below.

Acronym	**Data Block length**	**Key lengths**	**Number of Rounds**
AES	128 bits	128 bits 192 bits 256 bits	10 12 14
DES	64bits	56 bits (+8 bits of "checksum")	16

Because of its small key size, DES is sometimes run three times consecutively (Triple DES), for a total of 48 rounds. Triple DES has several variants, using either two or three keys.

There are many other encryption algorithms, but these are two of the most commonly encountered algorithms.

Public key (asymmetric) cryptography

Public key cryptography has two main uses:

1. Encrypt keys for use in symmetric cryptographic algorithms. Public key algorithms tend to be too slow for bulk encryption, so they are used to encrypt keys for transfer, which will then be used to encrypt bulk data.
2. Digitally sign/verify messages.

Public key cryptosystems generally have a pair of keys: a private key which must be kept secret, and a public key which is not sensitive and can be freely distributed.

There are two classes of public key algorithm commonly encountered.

1. Algorithms based on the difficulty of factoring
2. Algorithms based on the difficulty of the discrete log problem.

RSA

The security of RSA is based on the difficulty of factoring a modulus which is the product of two large random primes.

- **Private key:** Large primes p and q, typically on the order of 512-2048 bits each, and a decryption/signing exponent d = (1/e) mod (p-1)(q-1)
- **Public key:** Modulus N = p*q, and encryption/verification exponent e. e is typically 3, 17, or 65537.

The following are simplified[1] descriptions of how the RSA algorithm is used for encryption/decryption as well as for signing/verification.

Encryption/decryption

- **Encryption:** Given a message M, compute ciphertext C by $C = M^e \bmod N$.
- **Decryption:** To decrypt ciphertext C, compute $M = C^d \bmod N$.

Signing/verification

To sign a message M, let H(M) denote the hash of M (see below for info on hash functions).

- **Signature:** M is signed by computing S by $S = H(M)^d \bmod N$.
- **Verification:** Signature S for M is verified by computing H(M) and $H'(M) = S^e \bmod N$ and checking that H'(M) = H(M).

[1] In actual usage, a variety of padding schemes are used to pad the message M or hash(M) to the length of the modulus, before performing the private key operation.

RSA private key calculations are often sped up by performing calculations mod p and mod q separately, and combining the results using the Chinese Remainder Theorem (CRT).

RSA decryption with CRT

Suppose a message M has been encrypted as above: $C = M^e \bmod N$. Then using decryption exponents $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$, decryption with CRT is as follows:

1. $M_p = C^{dp} \bmod p$
2. $M_q = C^{dq} \bmod q$
3. $h = q^{-1} * (M_p - M_q) \bmod p$
4. $M = M_q + h*q$

Using the CRT can significantly speed the decryption process, and is commonly used where power and time considerations are critical.

Discrete log Based algorithms

The "classic" discrete log based algorithms are defined over the multiplicative group Z*(p) for large primes p. The discrete log problem is the following:

> In the group Z*(p), select a base point g. Then given a random point $h = g^a \bmod p$, find the exponent a (i.e. find the log of h to the base g, mod p).

In discrete log based algorithms, the secret is the exponent a, and the public value is $g^a \bmod p$. Several discrete log based algorithms commonly used are described below.

Diffie-Hellman (DH) key agreement protocol

Alice and Bob generate a shared secret key as follows:

1. Alice and Bob agree on a large prime p
2. Alice generates random secret a, and computes $h_a = g^a \bmod p$
3. Bob generates a random secret b, and computes $h_b = g^b \bmod p$
4. Alice and Bob exchange $h_a = g^a \bmod p$ and $h_b = g^b \bmod p$. (The exchange can occur on an unprotected channel.)
5. Alice and Bob compute shared secret $s = (h_a)^b \bmod p = (h_b)^a \bmod p = g^{ab} \bmod p$.

Digital Signature Algorithm (DSA)

This algorithm is a NIST standard for digitally signing messages. To sign a message M, let H(M) denote the hash of M.

- **Parameters:** Large primes $q < p$, where q divides p-1, and a number g where $g^{q-1} \bmod p = 1$

- **Key generation:**

  1. Alice generates a random secret key a, and computes public key $h = g^a \bmod p$

- **Signature:**

  1. Alice generates a random secret nonce k, and computes $r = (g^k \bmod p) \bmod q$
  2. Alice computes $s = k^{-1} * (H(M) + a * r) \bmod q$
  3. Signature of M is (r, s)

- **Verification**

  1. Bob computes $u_1 = s^{-1} * H(M) \bmod n$ and $u_2 = = s^{-1} * r \bmod q$
  2. $v = ((g^{u1} \cdot h^{u2}) \bmod p) \bmod q$
  3. Signature is valid if and only if v = r.

Other variants sometimes used include the Nyberg-Rueppel (NR), and Schnorr signature algorithms. They are similar computationally and rely on a secret exponent a and public value $g^a \bmod p$. See link below for details.

Elliptic curve cryptography

The algorithms above were defined over the multiplicative group Z*(p) for primes p. Elliptic curves are groups for which the discrete log problem is thought to be harder than Z*(p). A description of elliptic curves is outside the scope of this document. However, the Wikipedia article linked below gives a brief introduction.

From the perspective of the cryptographic algorithms above, the change from the group Z*(p) to an elliptic curve group is minimal. In particular, some of the exponentiations g^k is replaced with an elliptic curve point multiplications k*G, where k is the multiplier and G is a point on the elliptic curve. However, the flow of the algorithms is nearly identical for Z*(p) and elliptic curve groups.

Hash Functions

Hash functions are algorithms designed to transform arbitrary sized input data into a fixed length "fingerprint" for the data. Good hash functions have the following properties:

1. Preimage resistance: Given a hash value h, it should be hard to find a message M such that hash(M) = h.
2. Collision resistance: It should be hard to find two messages M_1 and M_2 with the same hash value (i.e. $hash(M_1) = hash(M_2)$).

Many hash functions operate by partitioning the input data into N-bit blocks, operating on each block in a series of rounds, with the output of one set of rounds chained to the input block to the next set of rounds. As with encryption algorithms, hash function round transformations typically operate on smaller sub-blocks of the N-bit block. For example, SHA-1 and SHA-256 operate on 32-bit words, while SHA-512 operates on 64-bit words.

The Secure Hash Algorithm (SHA) family of hash functions is one of the most widely used. There are a number of variants, shown below.

Acronym	**Data Block length**	**Output length**	**Number of Rounds**
SHA-1	512 bits	160 bits	80
SHA-256	512 bits	256 bits	64
SHA-512	1024 bits	512	80
SHA-384 (truncated SHA-512)	1024 bits	384	80

References

Acronym	Meaning	Link
AES	Advanced Encryption Standard	http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
CRT	Chinese Remainder Theorem	http://en.wikipedia.org/wiki/Chinese_Remainder_Theorem
DES	Data Encryption Standard	http://en.wikipedia.org/wiki/Data_Encryption_Standard
DH	Diffie-Hellman Key Exchange	http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
DSA	Digital Signature Algorithm	http://en.wikipedia.org/wiki/Digital_Signature_Algorithm
ECC	Elliptic Curve Cryptography	http://en.wikipedia.org/wiki/Elliptic_curve_cryptography
ECDH	Elliptic Curve Diffie-Hellman	http://en.wikipedia.org/wiki/Elliptic_curve_Diffie%E2%80%93Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm	http://en.wikipedia.org/wiki/ECDSA
ECNR	Elliptic Curve Nyberg Rueppel	http://www.cryptopp.com/wiki/Elliptic_Curve_Nyberg_Rueppel
ElGamal	Key exchange algorithm	http://en.wikipedia.org/wiki/ElGamal_encryption
HMAC	Hash-based Message Authentication Code	http://en.wikipedia.org/wiki/HMAC
MAC	Message Authentication Code	http://en.wikipedia.org/wiki/Message_authentication_code
RSA	Rivest-Shamir-Adleman	http://en.wikipedia.org/wiki/RSA_%28algorithm%29
SHA-1	Secure Hash Algorithm-1	http://en.wikipedia.org/wiki/SHA-1
SHA-2	Secure Hash Algorithm-2	http://en.wikipedia.org/wiki/SHA-2
Triple-DES	Triple Data Encryption Algorithm	http://en.wikipedia.org/wiki/Triple_DES