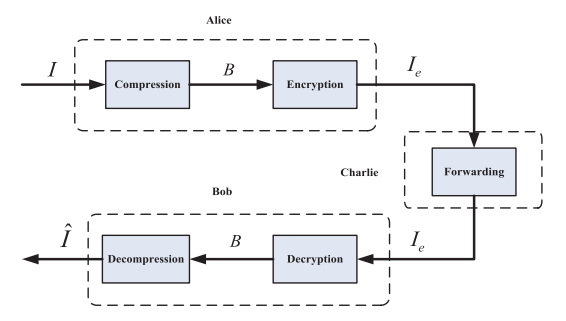
# Diseñando un sistema de encriptado y compresión via predicción de error de clustering y permutación aleatoria

**T8**

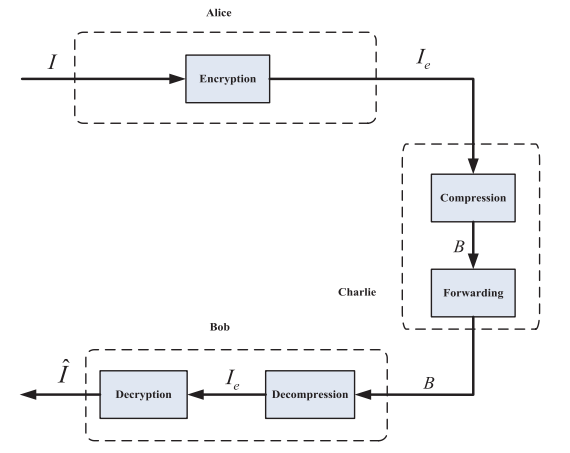


Pongamos por ejemplo que Alice quiere enviar una imagen I a Bob por medio de un canal inseguro que provee Charlie. Lo normal es comprimir y encriptar Ie con una función de encriptado Ek donde k es una clave secreta. Los datos Ie se pasan a Charlie y este lo manda a Bob. Cuando este lo recibe desencripta y luego descomprime.

La compresión y encriptado (CTE) tiene un orden que se debe invertir a veces. Alicia quiere proteger sus datos pero tiene unos recursos computacionales limitados como un móvil y no los puede malgastar comprimiendo. En contraste el proveedor del canal quiere comprimir todo el tráfico de la red para maximizar su uso por lo que está claro que lo mejor es delegar la tarea de compresión en Charlie que tiene mas recursos.

Un gran reto para el framework (ETC) encriptado y compresión es que la compresión se hace en el dominio encriptado y Charlie no tiene acceso a la clave k.

**T9**



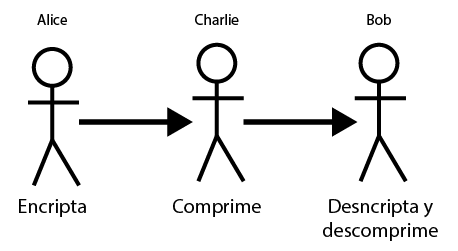
Charlie no va a poder comprimir los datos encriptados porque no hay ninguna estructura que pueda utilizar un compresor tradicional.

Los sistemas ETC actuales tienen una compresión baja comparándolos con los codificadores para entrada no encriptada.

Para el diseño del sistema se utilizarán imágenes en escala de grises, se propone una aproximación basada en la permutación en el dominio del error. Una codificación adaptativa al contexto (AC) puede comprimir los datos eficientemente. Hay una penalización <0.1% en la compresión. Debido a la alta sensibilidad de la secuencia de predicción de errores se mantiene un nivel de seguridad alto.

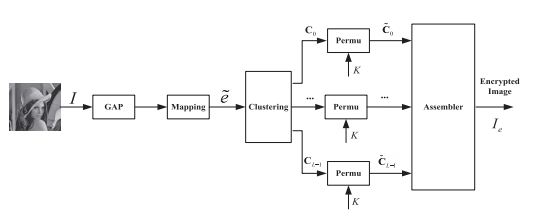
**T10**

**Sistema ETC propuesto**



El algoritmo de compresión debe considerar la seguridad de los datos. Se propone un esquema de encriptado que opera en el dominio de predicción de error.

**T11**



Para cada pixel Ii,j de la imagen la predicción se hace con un predictor de imagen GAP o MED. Se adopta el método GAP por su capacidad de decorrelación. El resultado se refina con un mecanismo de feedback adaptativo al contexto. El error se puede calcular como:

Para imágenes en 8 bits el error de predicción ei,j puede variar entre -255 y 25, se mapea en un rango de 0 a 255 considerando el valor predicho Iij que está disponible en el lado del decodificar, este valor también se utiliza para revertir el mapeado.

Se dividen los errores de predicción en L bloques (clusters) basados en una aproximación de contexto adaptativa. La aleatorización y compresión se beneficián de esta operación.Para indicar las actividades locales de la imagen se utiliza un estimador de energía de error. Para cada pixel es:

En un proceso de prueba se emplea una técnica de programación dinámica para conseguir un cluster óptimo en un sentido de entropía mínimo esta se minimiza:

El termino h denota la entropía de la secuencia de error de predicción en el cluster y una aproximación del bitrate que representa todos los errores de predicción. El cluster tiene una capacidad de compresión óptima.

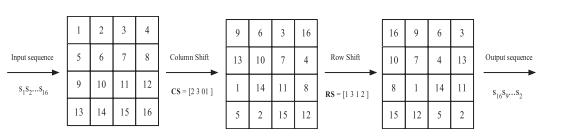
El parámetro de selección L necesita balancear la seguridad y la complejidad de compresión. A mayor tamaño mayor seguridad. Se obtiene un valor de 16 heurísticamente. El índice k de cada cluster se calcula como:

**T12**

El algoritmo va así:

1. Computar todas las predicciones de error de la imagen.
2. Dividirlos en L clusters, cada Ck se forma concatenando las predicciones de error mapeadas en orden de raster.
3. Redimensiona estos errores en cada Ck en un bloque 2D teniendo 4 filas y columnas donde |Ck| es el número de predicciones de error Ck.
4. Hacer dos operaciones des desplazamiento key-driven cíclicas para cada bloque y leer los datos en orden raster para obtener el cluster permutado.

CSk y RSk son los vectores clave secreta que controlan los offsets de columna y fila. Se obtienen de un flujo de llave generado por un cifrador que implica que los vectores pueden ser diferentes entre sesiones.

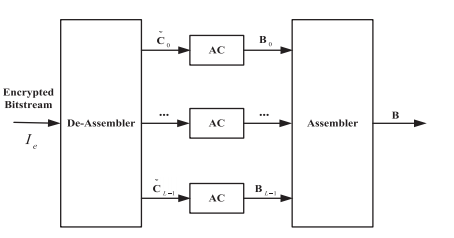


Se hace una permutación aleatoria. La columna se desplaza por un vector clave CS y se le aplica un desplazamiento de acuerdo a la clave asociada a cada columna. Este proceso se repite por filas. Los desplazamientos son circulares y fáciles de implementar.

1. El ensamblador concatena los clusters permutados y genera una imagen encriptada.
2. Se pasan los datos a Charlie. Los valores Ck hacen que pueda dividir Ie en L clusters

**T13**

**Compresión sin perdidas por AC adaptativo**

****

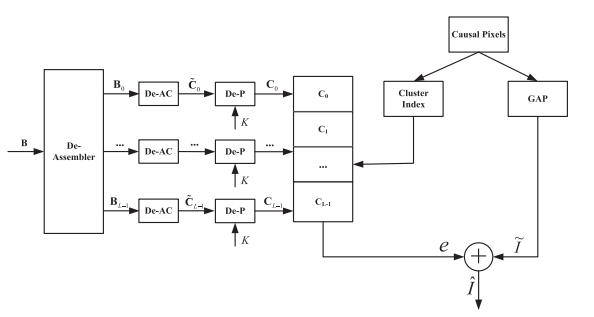
Charlie no tiene acceso a la clave secreta k y tiene que comprimir en el dominio encriptado.

Se utiliza un desensamblador que parsea Ie en L elementos como en la parte de encriptado. Un AC adaptativo codifica la secuencia de predicción sin perdidas en un flujo Bk. Esto se puede hacer en paralelo. Ahora, un ensamblador concatena todos los Bk para hacer el flujo comprimido B.

Se le tiene que enviar el tamaño de B a Bob. La compresión ha cambiado las posiciones pero no los valores de predicción. Así se preserva la función de probabilidad de la secuencia de predicción de error que controla el AC. El tamaño del flujo se calcula con:

T14

**Desencriptado y descompresión secuencial**



Bob divide B en L segmentos Bk, cada uno asociado a un cluster de error de predicción. Para cada Bk se aplica un decodificador adaptativo aritmético para obtener la secuencia de error permutada. Como sabe la clave K se puede obtener el Ck original.

Se decodifican los pixels en orden raster. Para cada Iij el estimador de energía de error y el valor predicho se pueden calcular con redondeos causales que ya han sido codificados. El primer error de predicción se selecciona y utiliza para derivar eij de acuerdo con iij y la regla de mapeo anterior. El pixel se computa como:

No hay distorsión en la predicción del error lo que implica decodificación sin errores.

**T15**

**T16**

