Diseño de un conjunto de herramientas software para ataques por canal lateral

Alberto Fuentes Rodríguez[1], Luis Hernández Encinas[1], Agustín Martín Muñoz[1] y Bernardo Alarcos Alcázar[2]

[1] *Departamento de Tratamiento de la Información y Criptografía (TIC)*
Instituto de Tecnologías Físicas y de la Información (ITEFI)
Consejo Superior de Investigaciones Científicas (CSIC)
{alberto.fuentes, luis, agustin}@iec.csic.es

[2] *Departamento de Automática*
Escuela Politécnica Superior
Universidad de Alcalá (UAH)
bernardo.alarcos@uah.es

29 de octubre de 2013

Índice

1. Introducción
   - Dispositivos criptográficos
   - Riesgos de las implementaciones
2. Ataque por análisis diferencial de potencia
3. Diseño de la herramienta
   - Objetivos de la herramienta
   - Representación de los puntos de la traza
   - Representación de la traza orientada a objetos
   - Resultados, conclusiones y trabajo futuro

¿Qué es un dispositivo criptográfico?

- Capacidad de cálculo.
- Los valores almacenados en la memoria no pueden ser modificados por el usuario.
- Realiza operaciones criptográficas.

Seguridad de los dispositivos criptográficos

- El usuario interactúa con el dispositivo mediante comandos.
- El procesador modifica/lee/procesa los valores internos...
- ...y devuelve un comando al usuario.

Presunción

Mientras el procesador lee, modifica o procesa los valores internos, los usuarios no pueden dar órdenes ni obtener información de los datos o de las operaciones.

Presunción

La seguridad está garantizada por la fortaleza matemática de los algoritmos criptográficos empleados.

Ataques por canal lateral

Los ataques por canal lateral rompen las anteriores presunciones.

Se puede escapar información por varios *canales laterales*:

- Potencia consumida.
- Campo electromagnético radiado.

Hipótesis

Las magnitudes medidas a través de los canales laterales dependen directamente de las instrucciones, de las operaciones matemáticas efectuadas y de los datos manejados por el procesador durante las operaciones criptográficas.

Ataques por canal lateral

- No invasivos.
- Pasivos.
- No se requiere conocimiento del dispositivo.
- Se puede sacar mucho provecho del conocimiento del algoritmo que está siendo ejecutado.

Análisis simple de potencia (SPA, *Simple Power Analysis*)

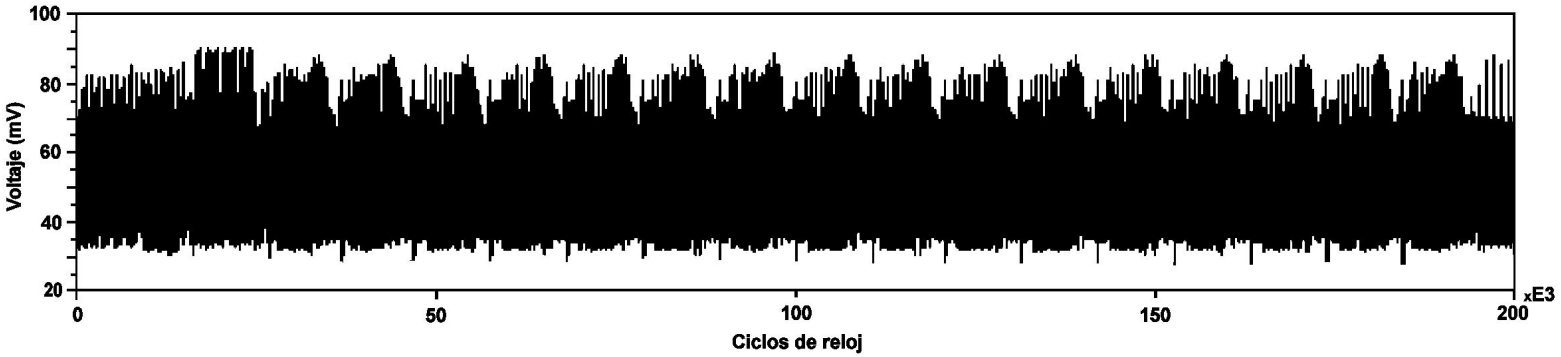


Figura: Consumo de potencia durante el cálculo con un DES

Análisis SPA de un RSA sin contramedidas

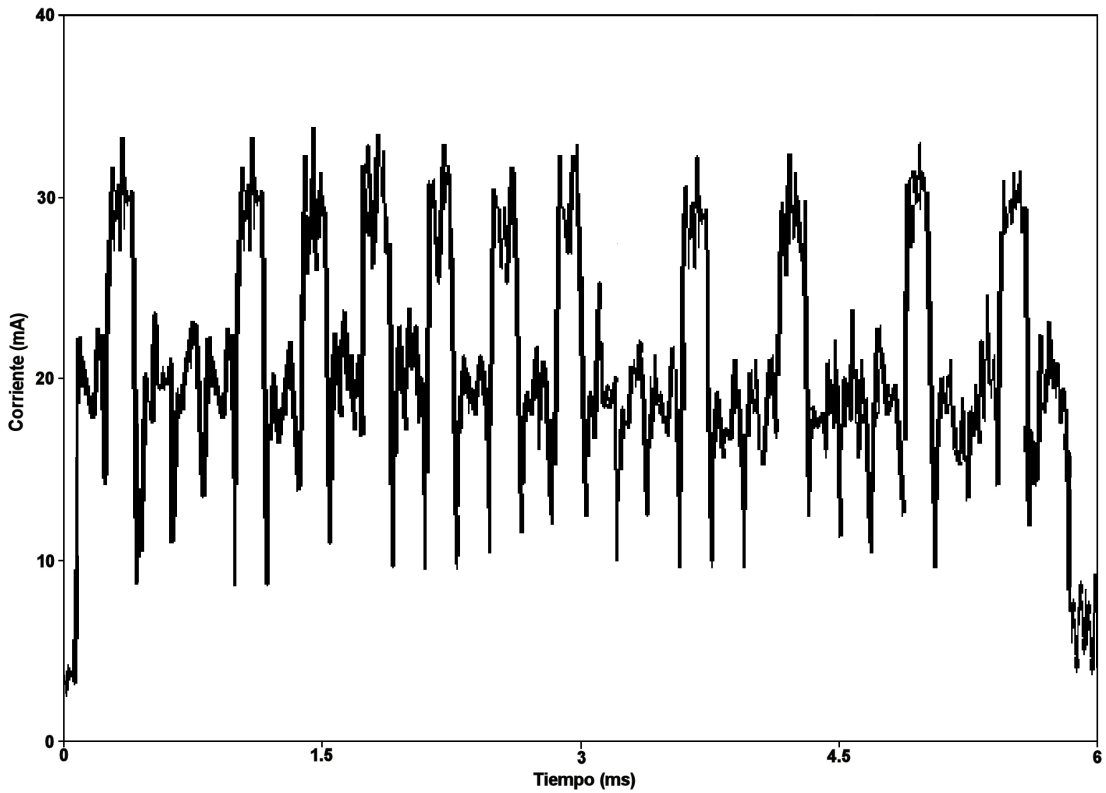


Figura: Consumo de potencia durante el cálculo con un RSA

Análisis SPA de un RSA sin contramedidas

Exponenciación modular: $y = x^k \pmod{n}$.

Algoritmo de elevar al cuadrado y multiplicar, utilizando la representación binaria de la clave $k = (k_{r-1}k_{r-2}\ldots k_0)$ y procesando los bits de izquierda a derecha:

1. $y = 1$.
2. For $i = (r-1)$ to 0 do:
   1. $y = y^2 \pmod{n}$.
   2. If $(k_i = 1)$ then $y = (y \cdot x) \pmod{n}$.
3. Return (y).

Ejemplo: Si la clave es $k = 23$, 10111 en binario, se obtiene:

$$a^{23} = (((a^2)^2 a)^2 a)^2 a.$$

Análisis SPA de un RSA sin contramedidas

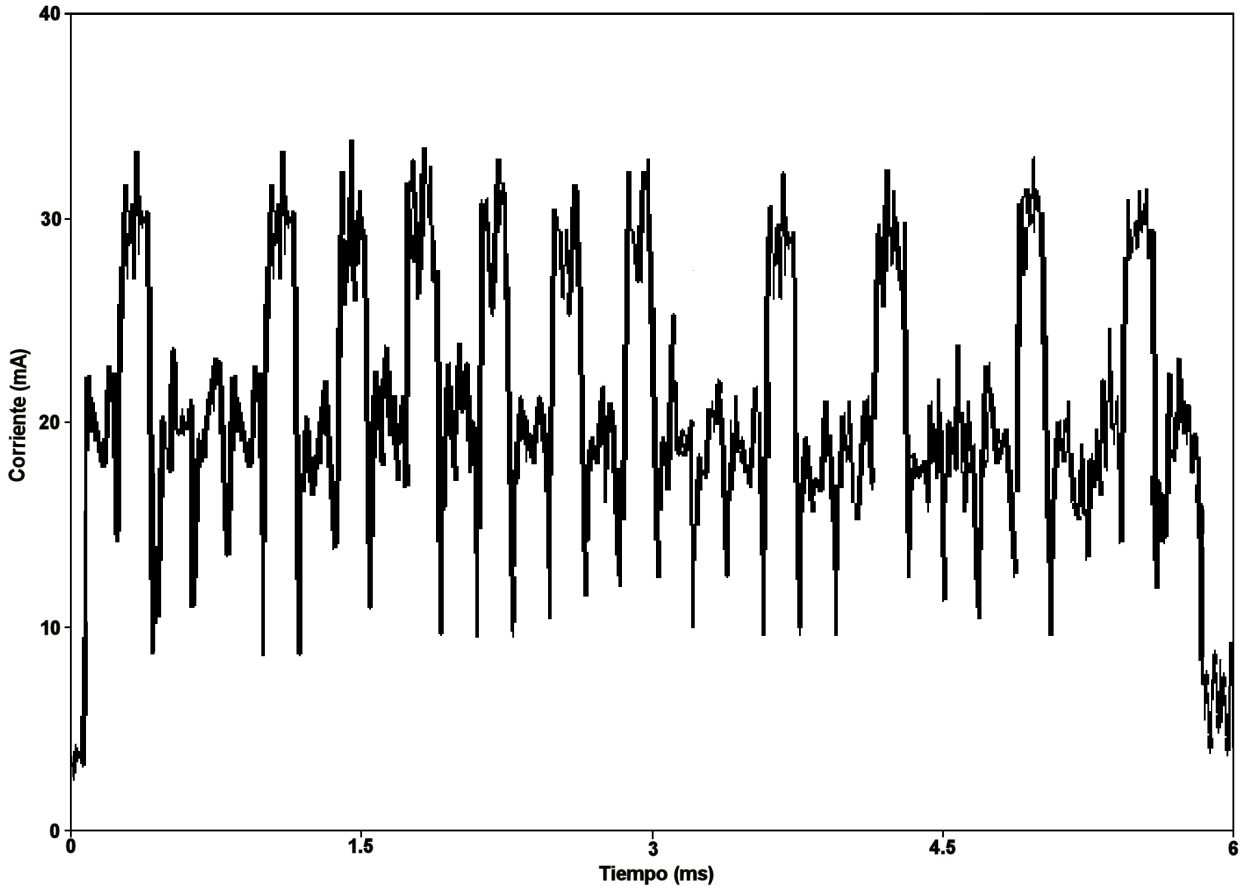


Figura: Consumo de potencia durante el cálculo de un RSA

Análisis SPA de un RSA sin contramedidas

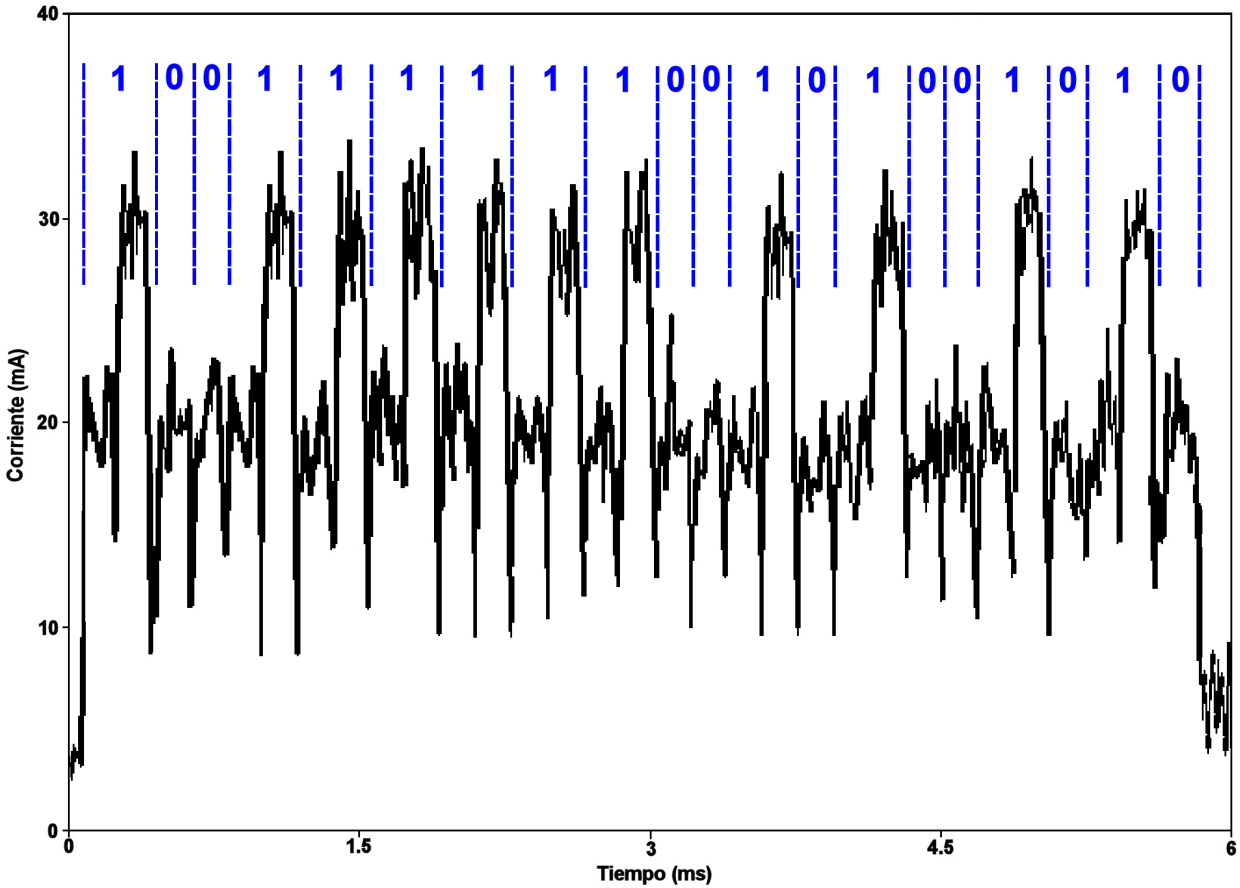


Figura: La clave es $k = 653642$

Fases de un DPA (*Differential Power Analysis*)

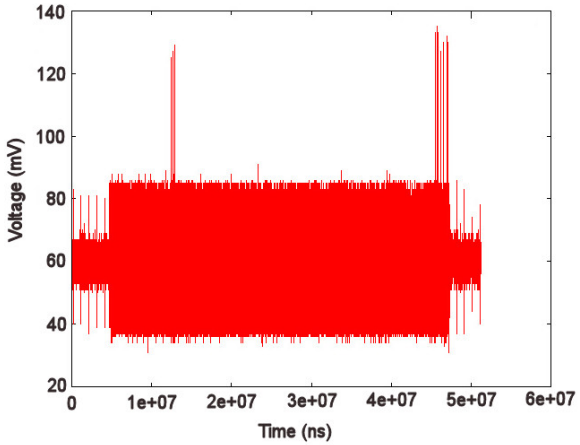
1. Elegir un paso intermedio del algoritmo criptográfico ejecutado cuya salida sea función de una parte de la clave y de datos conocidos (i.e. texto plano).
2. Ejecutar el algoritmo para diferentes valores de entrada, medir el consumo de potencia del dispositivo para cada entrada y almacenar los resultados en *trazas de potencia*.
3. Hacer diferentes hipótesis sobre la clave y calcular los correspondientes valores intermedios (valores calculados hipotéticos). Asociar estos valores calculados con un consumo hipotético, simulando el consumo mediante diferentes modelos.
4. Comparar mediante algoritmos estadísticos los consumos hipotéticos así obtenidos con el conjunto medido de trazas.

Objetivos de la herramienta

- Multiplataforma.
- Orientada a objetos (cualquier nuevo algoritmo se puede implementar fácilmente).
- Eficiente en cuanto a capacidad de cálculo y almacenamiento.

Resolución de almacenamiento

- La precisión del osciloscopio es de 8, 12, 16 bits **(valores en bruto)**
- La precisión flotante es de 32 o 64 bits **(valores reales)**



$$V = S \cdot \frac{Raw}{Max} + O$$

S Máximo de sensibilidad vertical (i.e. 50mv, 100mv, 200mv)

Raw Valor en bruto.

Max Valor máximo de la resolución.

O Offset DC.

Raw vs. *Float*

Ventajas de cada representación:

- Representación *Raw*
  - Se necesita menos memoria y menos espacio en disco.
  - Se optimizan los accesos a la caché.
- Representación *Float*
  - No se necesita operación de conversión.

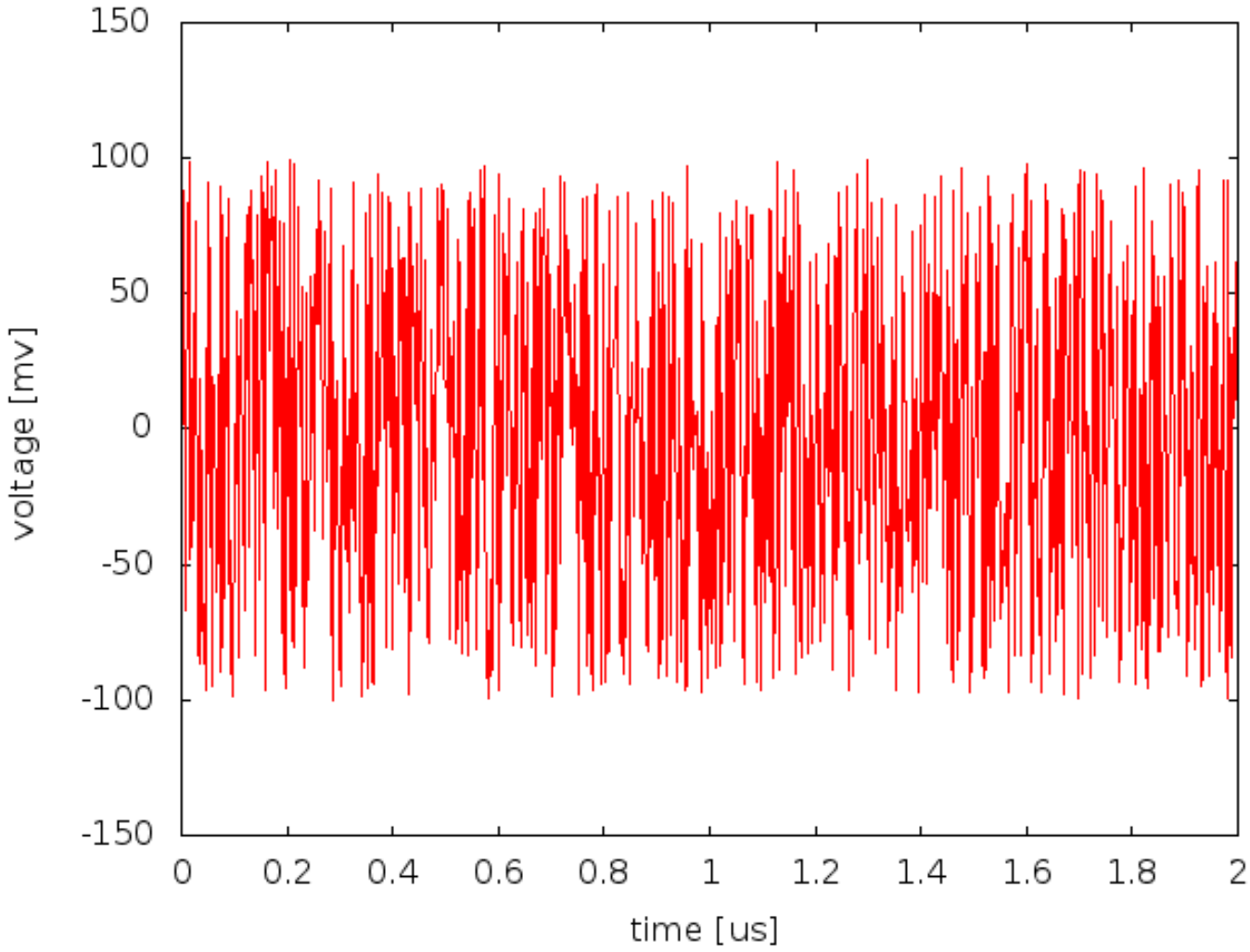
Opciones de la representación orientada a objetos

Plantillas C++ Clase genéricas *Trace*. El tipo de entrada especifica la resolución (por ejemplo, uint8_t para 8 bits).

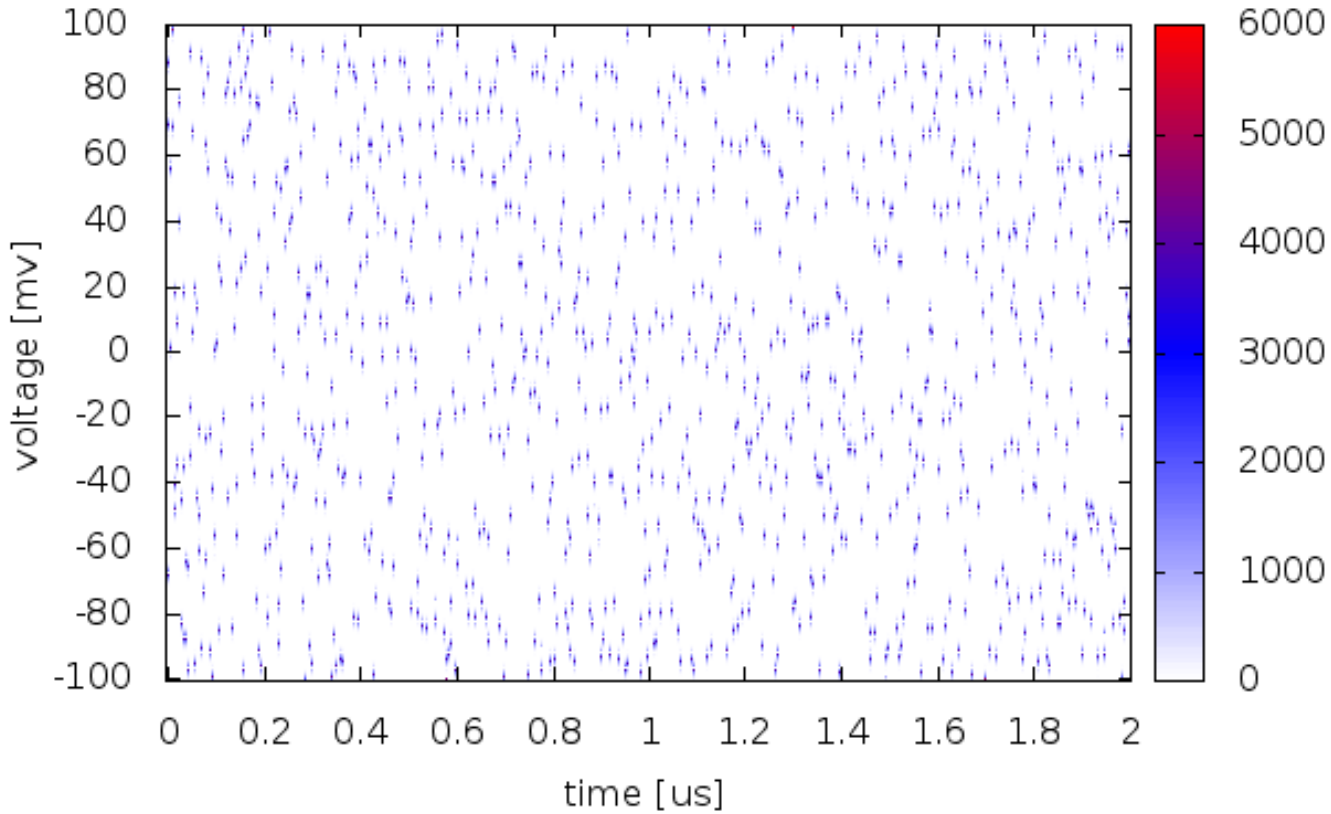
Herencia OO Clase abstracta *Trace*. Esta clase tiene una clase derivada para cada resolución.

Representación Float No requiere abstracción. La conversión de *raw* a valores de voltaje se hace sólo una vez.

Gráfica proporcionada por CTrace



Gráfica proporcionada por CTraceSet



Plantillas C++

Ventajas

- Sólo se requiere una clase.
- Resuelta en tiempo de compilación.
- Necesita menos memoria y menos disco.
- Optimiza el uso de la caché.

Inconvenientes

- Las plantillas se extienden por toda la herramienta.
- No se pueden separar fácilmente en un fichero cabecera (.h) y un fichero fuente (.cpp).
- Código fuente menos claro.

Herencias OO

Ventajas

- Abstracción al nivel de CTrace.
- Mecanismo bien conocido. Código más claro.
- Clases de bajo nivel simples. Pocas modificaciones futuras.
- Optimización de la caché.

Inconvenientes

- Es necesario crear una clase por resolución.
- Resuelta en tiempo de ejecución.
- Parte del tiempo se emplea en el envío dinámico.
- Requiere más tiempo de cálculo que las plantillas.

Representación flotante

Ventajas

- La conversión se hace sólo una vez.
- No se requiere abstracción.
- Simplicidad del código fuente.

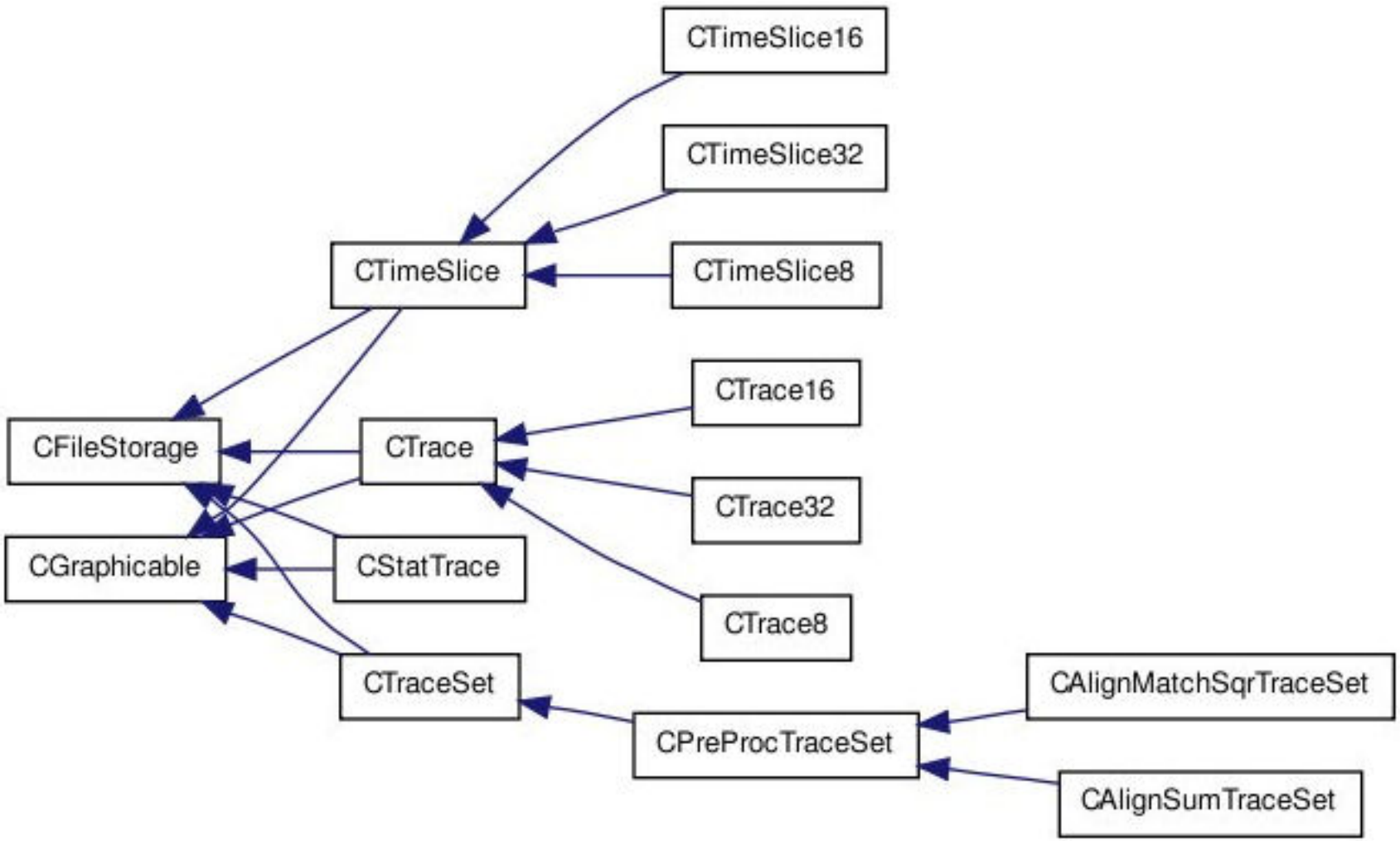
Inconvenientes

- Cada valor requiere 32 bits.
- Menos elementos en la memoria caché.

Comparación Estática vs. Dinámica

	Tiempo (s)			
	Optimización `-O0`		Optimización `-O3`	
Elementos	Estática	Dinámica	Estática	Dinámica
10 000 000	0.177548	0.189265	0.135924	0.120644
100 000 000	1.776202	1.891002	1.333785	1.201456

Jerarquía de clases



¡¡ Muchas gracias por su atención !!