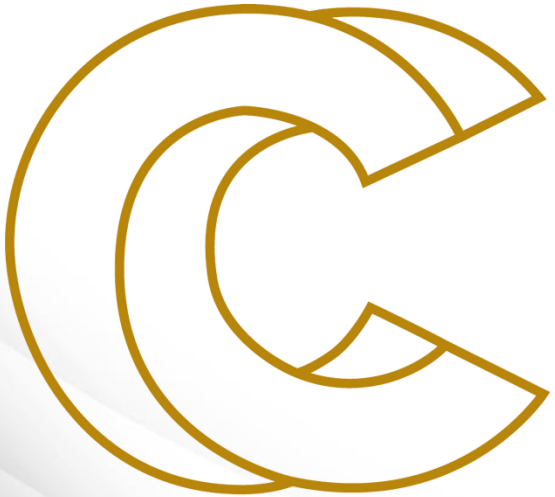


EURO 4SEE

GPU Assisted Brute Force Cryptanalysis of GPRS, GSM, RFID, and TETRA

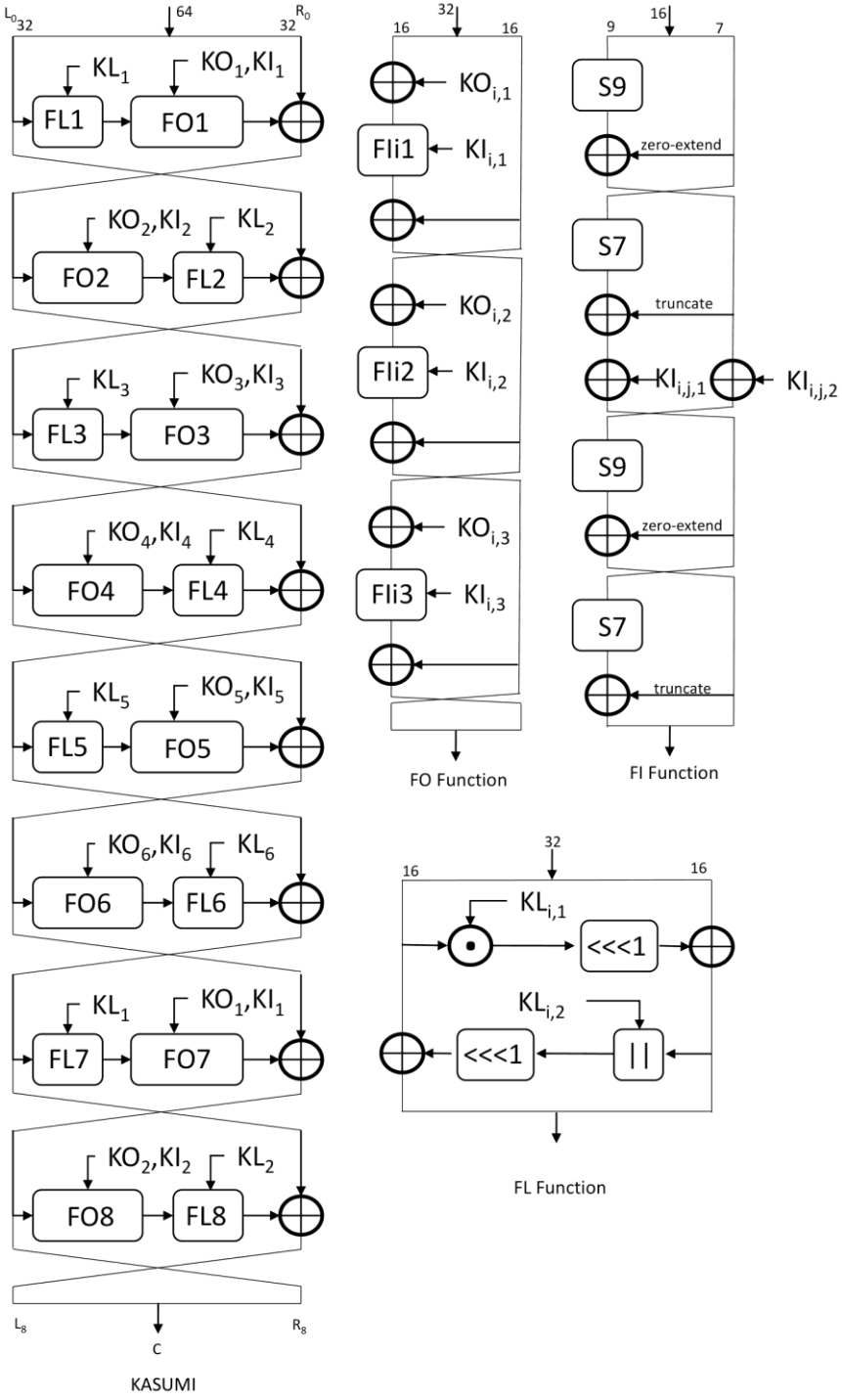
Cihangir Tezcan, PhD
Graduate School of Informatics, METU, Ankara

ncc@ulakbim.gov.tr

Lesson 7: CUDA Optimization of KASUMI

```
bit32 threadIndex = blockIdx.x * blockDim.x + threadIdx.x;
bit16 k1 = threadIndex/65536, k2 = threadIndex % 65536;

// Round 1
KL1 = LeftShiftd(k1, 1);
KL2 = k3 ^ 0x89AB;
KO1 = LeftShiftd(k2, 5);
KO2 = LeftShiftd(k6, 8);
KO3 = LeftShiftd(k7, 13);
KI1 = k5 ^ 0xFEDC;
KI2 = k4 ^ 0xCDEF;
KI3 = k8 ^ 0x3210;
temp = FLd(in_left, KL1, KL2);
temp = FOd(temp, KO1, KO2, KO3, KI1, KI2, KI3);
temp ^= in_right;
in_right = in_left;
in_left = temp;
```



Lesson 7: CUDA Optimization of KASUMI

CUDA Optimizations

1. We stored the two S-boxes ***S7*** and ***S9*** of KASUMI in the shared memory. Removing the bank conflicts did not provide speed-up.
2. The round constants of KASUMI can be kept in the shared memory, or in registers, or they can be provided as they are inside the code. Best performance was obtained when they are assigned to registers at the beginning of the kernel.
3. Our implementation of KASUMI for TMTO table creation and performing exhaustive key search are different because in an exhaustive search we rarely perform the encryption of the last round of a Feistel cipher.

Lesson 7: CUDA Optimization of KASUMI

CUDA Optimization on Register Count

1. In modern GPUs, best occupancy is obtained when the blocks consist of 1024 threads when there is no other significant bottleneck.
2. However, this can be achieved when the kernel uses less than or equal to 64 registers per thread because a block can have at most 64K registers in modern GPUs.
3. When compiled with many different versions of CUDA SDK and choice of compute capabilities between 5.0 and 8.9, our TMTO codes always require less than 64 registers and our exhaustive search codes that conditionally performs the last round require always more than 64 registers.
4. This prevents us to call the kernel blocks with 1024 threads and when we use 512 threads in blocks, conditionally performing the last round provides negligible speed up compared to the version where we perform all of the eight rounds with 1024 threads.
5. Since the increase in the number of required registers were due to the compiler's optimizations, we forced the CUDA SDK to use at most 64 registers when compiling our codes by using the command ***-maxrregcount=64***
6. This way, we achieved 10% speed up compared to the 8-round encryption of the TMTO table creation codes.
7. When compiling our KASUMI implementations, not limiting the register count to 64 at the compile time would result in "***too many resources requested for launch***" error at the run time.

Lesson 7: CUDA Optimization of KASUMI

GPU	Cores	Clock Rate	CC	Architecture
GTX 860M	640	1020 MHz	5.0	Maxwell
GTX 970	1664	1253 MHz	5.2	Maxwell
MX 250	384	1582 MHz	6.1	Pascal
RTX 2070 Super	2560	1770 MHz	7.5	Turing
RTX 3090	*10496*	*1700 MHz*	*8.6*	*Ampere*
RTX 4090	16384	2550 MHz	8.9	Ada Lovelace
RTX 5090	*21760*	*2407 MHz*	*10.1*	*Blackwell*

Lesson 7: CUDA Optimization of KASUMI

GPU	TMTO Table Creation	Key Search
MX 250	$2^{29.54}$ encryptions/s	$2^{29.70}$ keys/s
GTX 860M	$2^{29.79}$ encryptions/s	$2^{29.88}$ keys/s
GTX 970	$2^{31.41}$ encryptions/s	$2^{31.54}$ keys/s
RTX 2070 Super	$2^{32.58}$ encryptions/s	$2^{32.72}$ keys/s
RTX 4090	$2^{35.56}$ encryptions/s	$2^{35.72}$ keys/s

Next lecture

CUDA Optimization of SPECK

EURO 4SEE

Thanks!

Co-funded by
the European Union

EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101191697. The JU receives support from the Digital Europe Programme and Germany, Türkiye, Republic of North Macedonia, Montenegro, Serbia, Bosnia and Herzegovina.