

# GPU-Based Brute Force Cryptanalysis of KLEIN

Cihangir Tezcan<sup>a</sup>

*Department of Cyber Security, Graduate School of Informatics, Middle East Technical University, Ankara, Turkey*

**Keywords:** Cryptanalysis, Lightweight Cryptography, GPU, KLEIN.

**Abstract:** KLEIN is a family of lightweight block ciphers that supports 64-bit, 80-bit, and 96-bit secret keys. In this work, we provide a CUDA optimized table-based implementation of the KLEIN family which does not contain shared memory bank conflicts. Our best optimization reach more than 45 billion 64-bit KLEIN key searches on an RTX 4090. Our results show that KLEIN block cipher is susceptible to brute force attacks via GPUs. Namely, in order to break KLEIN in a year via brute force, one needs around 13, 1.34 million, and 111 billion RTX 4090 GPUs for 64-bit, 80-bit, and 96-bit secret keys, respectively. We recommend lightweight designs to avoid short keys.

## 1 INTRODUCTION


The Advanced Encryption Standard (AES) (Daemen and Rijmen, 2002) is arguably responsible for most of the encrypted data and after more than 20 years of cryptanalysis efforts, it is still secure against all known cryptanalysis techniques. Although AES is suitable and optimized for many platforms and use cases, resource-constrained devices might benefit from different encryption algorithms in terms of hardware size, latency, throughput, or battery consumption. Hence, many lightweight block ciphers were proposed for many different devices and platforms.

Security of modern ciphers does not depend on security by obscurity techniques. Instead, cipher designs are public and a well-designed cipher is secure as long as its secret key is generated randomly and kept secret. Thus, a well-designed encryption algorithm is resistant against non-generic attacks. Whereas generic attacks provide a security upper bound. For instance, regardless of the design of a cipher, an attacker can capture a plaintext and its corresponding ciphertext under a secret key and encrypt the plaintext with every possible key to check if the expected ciphertext is observed. Such an attack is called brute force or exhaustive key search attack. For a  $k$ -bit secret key, such an exhaustive search requires at most  $2^k$  encryptions. Thus, the key size  $k$  must be selected depending on the current and foreseeable future technology to prevent generic attacks. For instance, 112-bit secret keys are assumed by NIST (Barker and

Roginsky, 2019) to be secure until 2030 and maybe later. However, there are some ISO/IEC standard encryption algorithms that support 80-bit keys. Although AES key size is at least 128 bits, some of the lightweight designs use shorter keys for better performance. Yet short keys might make them susceptible to brute force attacks.

KLEIN (Gong et al., 2011) is an example for such a lightweight block cipher. It is software-oriented and it has AES-like design. However, unlike AES, KLEIN supports three short key sizes: 64-bit, 80-bit, and 96-bit. Thus, KLEIN can provide short term security due to its short key sizes. However, the length of this short term security depends on the current technology and it should be calculated so that the users might have an idea about how long their encrypted data will remain secret.

An exhaustive search attack is easily parallelizable since we are performing the same encryption operation with a different candidate key. To perform the computations, an attacker can use central processing units (CPUs), graphics processing units (GPUs), FPGAs, or ASICs. GPUs outperforms CPUs in parallelizable operations since they have thousands of cores and they have single instruction multiple data architecture. FPGAs can outperform GPUs especially when the operations are not memory intensive. Moreover, FPGAs might be energy efficient compared to GPUs but they require expertise and they are not as easily accessible as GPUs. Since ASICs are dedicated devices, they outperform FPGAs or GPUs but manufacturing costs must be considered since these devices

<sup>a</sup>  <https://orcid.org/0000-0002-9041-1932>

can only perform a specific function.

Exhaustive search attack implementation of a symmetric key encryption algorithm on a GPU can be categorized into three methods: Naive, table-based, and bitsliced. In a naive implementation, every operation of the encryption algorithm is implemented as they are. Table-based implementations aim to precompute and store outputs of layers of the cipher for every possible input. Thus, they can be regarded as a time-memory trade-off for a naive implementation.

Since the input space is large, table-based implementations partition the input space so that these partitions can be computed and stored independently and their results can be combined at the end. Such pre-computed tables are called T-tables. In GPU implementations, these tables are generally stored in the shared memory for better performance, instead of other memory types like global or constant memory. The bottleneck in this approach is the bank conflicts in the shared memory and the inability to use large T-tables due the limited shared memory size of GPUs.

In bitslicing technique every bit is kept in a different variable. This approach removes the operations that are needed to access a single bit in a byte or a larger data type. Bitsliced implementations are favorable when the state of the cipher is small and the cipher design contains operations on bits like in the case of CRYPTO1 (Tezcan, 2017). However, an efficient bitsliced GPU implementation is also provided in (Nishikawa et al., 2017) for AES.

Having a fast implementation of a cipher can be used for many purposes other than fast encryption. For instance, the current best GPU implementation of AES (Tezcan, 2021) is used in (Belorgey et al., 2023) as AES-CTR-based masking function in their aggregation protocol on the concept of counter-based cryptographically-secure pseudorandom number generators (csPRNGs), a concept that is used by Facebook in their *torchsprng* csPRNG. They improved upon *torchsprng* using the optimizations of (Tezcan, 2021) and obtained 100x speedup in the masking function compared to a single CPU core.

A fast implementation can be used to experimentally verify theoretically obtained results. Moreover, it also allows us to check the strength of the brute force attacks on short keys. For instance, key length of block ciphers were revisited in (Tezcan, 2022) where it was shown that 56-bit DES and 80-bit PRESENT secret keys are well within the reach of current GPU technology.

In this work we use the ideas of (Tezcan, 2021) which were used to remove shared memory bank conflicts in GPU optimizations of AES. Since KLEIN has AES-like structure, we obtained a shared mem-

ory bank conflict free optimization of KLEIN and we can try  $2^{35.40}$  64-bit KLEIN keys per second on an RTX 4090 GPU.

Security of KLEIN were analyzed against known cryptanalytic techniques and a full-round truncated differential attack on KLEIN with 64-bit was provided by (Lallemand and Naya-Plasencia, 2014). This attack requires  $2^{57.07}$  encryptions and  $2^{54.5}$  data. Time and data complexities of this attack was improved in (Rasoolzadeh et al., 2017) which now requires  $2^{54.9}$  encryptions and  $2^{48.6}$  data. Note that these attacks still require huge amount of encryptions and the authors of those attacks could not verify them in practice. Thus, having fast and optimized implementations are crucial for verification of theoretically obtained results. Note that it takes less than 8.5 days to verify the attack of (Rasoolzadeh et al., 2017) using our optimized codes on a single RTX 4090. With multiple GPUs, the verification can be done in hours.

Attacks slightly better than the exhaustive search on every three version of KLEIN were also obtained in the literature. A biclique attack on 64-bit version of KLEIN was provided in (Ahmadian et al., 2015) which requires  $2^{62.8}$  encryptions. Similarly, biclique attacks on all versions of KLEIN were provided in (Abed et al., 2012) which require  $2^{79}$  and  $2^{95.8}$  encryptions for the key sizes of 80 bits and 96 bits, respectively. Our GPU optimizations can be used to verify full or reduced versions of these attacks.

## 2 KLEIN

KLEIN is a software-oriented lightweight block cipher family that was proposed at RFIDSec 2011 (Gong et al., 2011). It has a compact implementation design and requires low memory both in hardware and software. This makes KLEIN suitable for resource-constrained devices like wireless sensors or RFID tags.

KLEIN is a Substitution-Permutation Network that works on blocks of 64 bits. It supports three key lengths  $k$ , namely 64, 80, and 96 bits and we denote these versions by KLEIN- $k$ . The number of rounds for these key lengths are 12, 16, and 20, respectively. A 64-bit state of KLEIN is represented by 16 nibbles. Each round consists of 4 layers:

1. Round key is XORed with the state.
2. A  $4 \times 4$  S-box is applied to the state 16 times in parallel.
3. The state is rotated two bytes to the left.
4. The state is divided into two parts and both of them are multiplied by the MDS matrix of AES.

Table 1: The specifications of the GPUs that are used in this work. CC denotes CUDA compute capability.

GPU	Cores	Clock Rate	CC	Architecture
MX 250	384	1582 MHz	6.1	Pascal
GTX 970	1664	1253 MHz	5.2	Maxwell
RTX 2070 Super	2560	1770 MHz	7.5	Turing
RTX 4090	16384	2550 MHz	8.9	Lovelace

Round keys are generated from the master key and it consists of XOR, swap, four S-box, and round constant XOR operations. A more detailed information for KLEIN can be found in (Gong et al., 2011).

In this work, our main aim is to optimize KLEIN on GPUs. We used many different GPUs from different architectures to show that our optimizations are not valid only for a specific GPU. The specifications of the GPUs that are used in this work are provided in Table 1.

### 3 CUDA OPTIMIZATION OF KLEIN

To the best of our knowledge, the best known GPU optimization of AES was provided in (Tezcan, 2021). It is a table-based implementation where the tables are kept in the shared memory of the GPU and due to a good arrangement of the tables, no shared memory bank conflicts occur when different threads in a warp try to read the same table value. Since KLEIN has an AES-like structure, it is desirable to use the same approach.

Although KLEIN also operates on bytes, its S-box works on nibbles instead of bytes. And if we create our tables according to nibbles, resulting table-based implementation will require more operations than AES and will be slower than AES. Thus, we combined every two consecutive  $4 \times 4$  S-box of KLEIN in order to turn them into an  $8 \times 8$  S-box. Then we created the tables by combining the three layers of the round function after the round key addition. Namely for each input of the  $8 \times 8$  S-box, we calculated the result of the S-box operation, two bytes to the left, and the matrix multiplication. The result can be stored in an array of 256 elements with 32-bit sizes. We need to create four tables in this respect due to the four bytes that are multiplied with the matrix. However, these tables turn out to be one byte rotations of each other, due to the choice of AES matrix. Thus, keeping a single table and obtaining the others by rotations are possible.

In current GPU architectures, threads work as warps which consists of 32 threads. And there are 32 data lanes these threads in a warp can use to access the shared memory. If two threads try to read values

that are in the same shared memory bank, these operations become serialized. In order to avoid shared memory bank conflicts, 32 copies of AES table are stored in (Tezcan, 2021) which allowed every thread in a warp to use its own data lane. Similarly, we calculated the table for KLEIN and stored it in the global memory of GPU as *T0G*. The following CUDA code writes that table to the shared memory 32 times to avoid shared memory bank conflicts.

```

bit32 threadIdx = blockIdx.x * blockDim.x +
    threadIdx.x;
int warpThreadIndex = threadIdx.x & 31;
__shared__ bit32 TOS[256][32];
if (threadIdx.x < 256)
    for (int i = 0; i < 32; i++) TOS[threadIdx.x][i]
        = T0G[threadIdx.x];
    
```

32 copies of this table requires 32KB of shared memory. Since current GPUs come with 48KB of shared memory, we cannot do this for the other three tables. Thus, we only use one table and obtain the rest by byte rotations. When we store the key with two variables *key1* and *key0* and the state as *plaintext1* and *plaintext0*, one round of encryption turns into the following CUDA code for KLEIN-64:

```

temp1 = plaintext1 ^ (key1 >> 16);
temp0 = plaintext0 ^ (key1 << 16) ^ (key0 >> 32);
plaintext0 = arithmeticRightShift(TOS[(temp1 & 0
    x00FF0000) >> 16][warpThreadIndex], 24) ^
    arithmeticRightShift(TOS[(temp1 & 0xFF000000)
    >> 24][warpThreadIndex], 16) ^
    arithmeticRightShift(TOS[temp0 & 0x000000FF][
    warpThreadIndex], 8) ^ TOS[(temp0 & 0x0000FF00
    ) >> 8][warpThreadIndex];
plaintext1 = arithmeticRightShift(TOS[(temp0 & 0
    x00FF0000) >> 16][warpThreadIndex], 24) ^
    arithmeticRightShift(TOS[(temp0 & 0xFF000000)
    >> 24][warpThreadIndex], 16) ^
    arithmeticRightShift(TOS[temp1 & 0x000000FF][
    warpThreadIndex], 8) ^ TOS[(temp1 & 0x0000FF00
    ) >> 8][warpThreadIndex];
    
```

Since NVIDIA GPUs do not have an instruction for bit rotations, we perform two shift and one XOR operation to perform the rotation which is denoted as *arithmeticRightShift()* in our codes. Although there is no single instruction for bit rotations, it was observed in (Tezcan, 2021) that CUDA's byte permutation instruction *\_\_byte\_perm* can be used in our calculations since our bit rotations are a multiple of 8.

Table 2: Number of key searches per second for the exhaustive key search attack on KLEIN.

GPU	KLEIN-64	KLEIN-80	KLEIN-96
MX 250	$2^{29.70}$ keys/s	$2^{00.00}$ keys/s	$2^{28.42}$ keys/s
GTX 970	$2^{31.75}$ keys/s	$2^{30.74}$ keys/s	$2^{30.48}$ keys/s
RTX 2070 Super	$2^{33.19}$ keys/s	$2^{32.46}$ keys/s	$2^{32.17}$ keys/s
RTX 4090	$2^{35.40}$ keys/s	$2^{34.74}$ keys/s	$2^{34.39}$ keys/s

Although using this instruction allows us to use a single instruction instead of three, apparently new generation GPUs like RTX 2070 Super and RTX 4090 perform the same operations in both cases because the change in the performance was negligible in our experiments. However, using the `_byte_perm` instruction provided 5% speedup on GTX 970.

Key schedule requires calculation of four  $4 \times 4$  S-boxes and we turned that into two  $8 \times 8$  S-box calculations. Since we can store this S-box as 8-bit *unsigned char* array instead of 32-bit *unsigned int*, we have enough shared memory and can store 32 copies of it to avoid shared memory bank conflicts. However, we observed that shared memory bank conflicts in reading these 8-bit S-box values does not cause the delays we observed for the bank conflicts for *TOS*. Thus, we got better occupancy on the GPU when we kept a single copy of this table in the shared memory.

Using our best optimizations<sup>1</sup>, we performed exhaustive key search attack on every version of KLEIN using many GPUs. The number of keys that we can try in a second are provided in Table 2.

Main difference between the performance of the three versions of KLEIN comes from the number of rounds of each version. Namely, 12, 16, and 20 rounds for 64-bit, 80-bit, and 96-bit secret keys. Moreover, our KLEIN-64 implementation is also faster than the other variants because the 64-bit secret key can be stored in two 32-bit unsigned integers. However, we had to use two 64-bit integers in our KLEIN-80 and KLEIN-96 implementations. Since GPU architectures are designed for 32-bit operations, 64-bit operations are slower.

Since the design of KLEIN is similar to AES and we used similar optimization techniques, we provide the performance of the exhaustive search attack on these two block ciphers on the same GPU in Table 3. Although KLEIN has more rounds than AES, it can be seen that our KLEIN optimization is faster than AES because our optimizations require less operations.

<sup>1</sup>Our table-based optimized KLEIN CUDA codes are publicly available at GitHub so that they can be used to verify our experiments, to analyze KLEIN, or to compare future optimizations: <https://www.github.com/cihangirtezca/n/CUDA.KLEIN>

Table 3: Number of key searches per second for the exhaustive key search attack on KLEIN and AES performed for different key sizes on a single RTX 2070 Super GPU.

Cipher	Keys/s		
AES - 128 / 192 / 256	$2^{32.43}$	$2^{32.01}$	$2^{31.66}$
KLEIN - 64 / 80 / 96	$2^{33.19}$	$2^{32.46}$	$2^{32.17}$

## 4 CRYPTANALYSIS OF KLEIN

Our key search results that are provided in Table 2 can be used to estimate how long will it take to perform brute force attacks on the three versions of KLEIN. A year consists of around  $2^{24.91}$  seconds. Thus, we can try  $2^{35.40+24.91} = 2^{60.31}$  KLEIN-64 keys per second on an RTX 4090 and capture the key in less than 13 years. Performing the same attack with a million RTX 4090 GPUs reduces the attack time to less than 5 days.

A biclique attack on KLEIN-64 was provided in (Ahmadian et al., 2015) which requires  $2^{62.8}$  encryptions and  $2^{39}$  data. Thus, we can perform this attack on a single RTX 4090 in less than 6 years.

A truncated differential attack on the full 12 rounds of KLEIN-64 was proposed in (Lallemand and Naya-Plasencia, 2014). That attack requires  $2^{57.07}$  encryptions and authors tried to experimentally verify their attack by using a C implementation on Intel(R) Xeon(R) CPU W3670 at 3.20GHz (12MB cache) with 8GB of RAM. However, it would take hundreds of years to complete the experiment on the 12-round attack. Instead, they performed their experiments on the reduced versions of their attack. When the attack is reduced to 10 rounds, the time complexity reduces to  $2^{44.4}$  encryptions and they performed it in 15 days. Similarly they performed their attack on 9 rounds which requires  $2^{38}$  encryptions and it took around 2 days.

The attacks of (Lallemand and Naya-Plasencia, 2014) perform partial encryptions and decryptions. A small modification to our optimized CUDA codes can be used to perform these attacks. It should be noted that such a modification would introduce a small overhead to the performance. Since we can perform  $2^{35.40}$  KLEIN-64 encryptions per second on a single RTX 4090, the 9-round experiment that requires  $2^{38}$  encryptions would take just a few seconds with our GPU



implementation. Similarly, the 10-round experiment that took 15 days when run on CPU would take less than 10 minutes with our proposed GPU optimizations. Moreover, the full 12-round attack that requires  $2^{57.07}$  encryptions would take  $2^{21.67}$  second which is less than 39 days. Note that it would take more than 300 years to verify this attack on the CPU setup and the C implementation of (Lallemand and Naya-Plasencia, 2014).

The attacks of (Lallemand and Naya-Plasencia, 2014) was improved in (Rasoolzadeh et al., 2017) which now requires  $2^{54.9}$  encryptions. Performing  $2^{54.9}$  encryptions would take less than 9 days with our CUDA codes on an RTX 4090.

Our optimization results show that we can try  $2^{34.74+24.91} = 2^{59.65}$  KLEIN-80 keys in a year. This means that it would take  $2^{20.35}$  years for an RTX 4090 to capture a KLEIN-80 or it would require  $2^{20.35} \approx 1.34$  million RTX 4090 GPUs to capture the key in a year.

A biclique attack in (Abed et al., 2012) has a time complexity of  $2^{79}$  encryptions which is two times faster than the exhaustive search attack. However, this attack requires  $2^{60}$  memory and implementing this attack using our GPU optimizations might result in an attack that is slower than the exhaustive search. Because storing and processing  $2^{60}$  data would introduce a significant overhead.

Our optimization results show that we can try  $2^{34.39+24.91} = 2^{59.3}$  KLEIN-96 keys in a year. This means that it would take  $2^{36.7}$  years for an RTX 4090 to capture a KLEIN-96 or it would require  $2^{36.7} \approx 111$  billion RTX 4090 GPUs to capture the key in a year.

A biclique attack in (Abed et al., 2012) has a time complexity of  $2^{95.18}$  encryptions which is  $2^{0.82}$  times faster than the exhaustive search attack. However, this attack requires  $2^{60}$  memory and implementing this attack using our GPU optimizations might result in an attack that is slower than the exhaustive search. Because storing and processing  $2^{60}$  data would introduce a significant overhead.

Although an exhaustive key search attack on GPUs does not look realistic with these numbers, it should be noted that this attack can become practical in the future since new GPUs are always built with more cores and faster clock speeds. Moreover, GPUs are general purpose devices and if an attack on 96-bit KLEIN becomes profitable, one can built ASICs where this attack becomes practical and requires less electricity than GPUs.

## 5 CONCLUSIONS

In this work we provided a CUDA optimized table-based implementation of the KLEIN family of block ciphers which does not contain shared memory bank conflicts. Our best optimization reach  $2^{35.40} \approx 45$  billion KLEIN-64 key trials on an RTX 4090. Our results show that KLEIN block cipher that supports 64-bit, 80-bit, and 96-bit secret keys is susceptible to brute force attacks via GPUs. Thus, lightweight designs should not support short keys.

## ACKNOWLEDGEMENTS

This work has been supported by The Scientific and Technological Research Council of Türkiye (TÜBİTAK) and German Academic Exchange Service (DAAD) Bilateral Research Cooperation Project (TÜBİTAK 2531 Project) under the grant number 123N546 and titled "Cryptanalysis of Symmetric Key Encryption Algorithms: Theory vs. Practice".

This project has also been supported by Middle East Technical University Scientific Research Projects Coordination Unit under grant number AGEP-704-2023-11294.

## REFERENCES

- Abed, F., Forler, C., List, E., Lucks, S., and Wenzel, J. (2012). Biclique cryptanalysis of present, led, and klein. *Cryptology ePrint Archive*, Paper 2012/591. <https://eprint.iacr.org/2012/591>.
- Ahmadian, Z., Salmasizadeh, M., and Aref, M. R. (2015). Biclique cryptanalysis of the full-round KLEIN block cipher. *IET Inf. Secur.*, 9(5):294–301.
- Barker, E. and Roginsky, A. (2019). Transitioning the use of cryptographic algorithms and key lengths. *NIST SP 800-131A Rev. 2*.
- Belorgey, M. G., Dandjee, S., Gama, N., Jetchev, D., and Mikushin, D. (2023). Falkor: Federated learning secure aggregation powered by AESCTR GPU implementation. In Brenner, M., Costache, A., and Rohloff, K., editors, *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Copenhagen, Denmark, 26 November 2023*, pages 11–22. ACM.
- Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer.
- Gong, Z., Nikova, S., and Law, Y. W. (2011). KLEIN: A new family of lightweight block ciphers. In Juels, A. and Paar, C., editors, *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected*

- Papers*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer.
- Lallemand, V. and Naya-Plasencia, M. (2014). Cryptanalysis of KLEIN. In Cid, C. and Rechberger, C., editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 451–470. Springer.
- Nishikawa, N., Amano, H., and Iwai, K. (2017). Implementation of bitsliced AES encryption on cuda-enabled GPU. In Yan, Z., Molva, R., Mazurczyk, W., and Kantola, R., editors, *Network and System Security - 11th International Conference, NSS 2017, Helsinki, Finland, August 21-23, 2017, Proceedings*, volume 10394 of *Lecture Notes in Computer Science*, pages 273–287. Springer.
- Rasoolzadeh, S., Ahmadian, Z., Salmasizadeh, M., and Aref, M. R. (2017). An improved truncated differential cryptanalysis of Klein. *Tatra Mountains Mathematical Publications*, 67(1):135–147.
- Tezcan, C. (2017). Brute force cryptanalysis of MIFARE classic cards on GPU. In Mori, P., Furnell, S., and Camp, O., editors, *Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017, Porto, Portugal, February 19-21, 2017*, pages 524–528. SciTePress.
- Tezcan, C. (2021). Optimization of advanced encryption standard on graphics processing units. *IEEE Access*, 9:67315–67326.
- Tezcan, C. (2022). Key lengths revisited: Gpu-based brute force cryptanalysis of DES, 3DES, and PRESENT. *J. Syst. Archit.*, 124:102402.

