THE AES-128 DECRYPTION CIRCUIT FOR TRANSCIPHERING IN TFHE

Max Tanis

INTRODUCTION

- Banks need fraud detection without exposing sensitive data
- Homomorphic Encryption (HE) allows computation on encrypted data, but causes massive ciphertext expansion
- Transciphering solves this by combining symmetric encryption with Fully Homomorphic Encryption (FHE)
- Fast Fully Homomorphic Encryption over the Torus (TFHE) encrypts each bit separately, useful for Boolean symmetric ciphers like AES-128

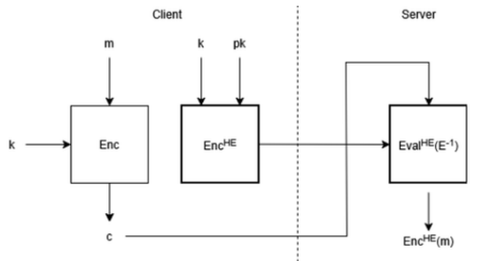

Figure 1: The transciphering protocol. The client encrypts their data (m) symmetrically and the symmetric key (k) homomorphically using public key (pk). The server evaluates the decryption circuit using homomorphic operations (thick-bordered boxes) without learning the plaintext.

RESEARCH QUESTION

How does the computational and communication overhead of homomorphically evaluating the AES-128 decryption circuit in TFHE scale with varying dataset sizes?

EXPERIMENT

SET UP

- Transciphering in TFHE-rs
- Boolean circuit evaluates the AES decryption circuit
  - 6,400 AND gates, each requiring bootstrapping
- Integrated into the TPSI framework

RUN

- Program is run on different dataset sizes
- Each size is run 30 times for reliability
- Metrics: communication and computational overhead
- Finally, compare AES-128 with an HE-friendly cipher, Kreyvium

RESULTS

COMPUTATIONAL OVERHEAD

- Init phase: ~72s, independent of dataset size
- Per-bit latency decreases from 916 ms/bit (S=10) to 371 ms/bit (S=500)

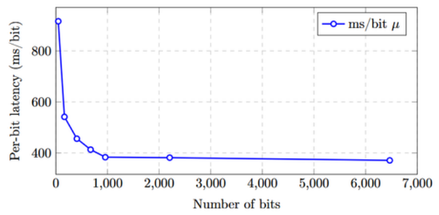

Figure 3: Per-bit latency of AES-128 transciphering vs number of bits. Data used from Table 1.

Table 1: Computational overhead of AES-128 transciphering per dataset size. With initialization time and per-bit latency as mean (μ) and standard deviation (σ) over 30 runs

S	# Bits	Init μ (s)	Init σ (s)	ms/bit μ	ms/bit σ
10	48	74.44	0.85	916.17	10.63
25	168	74.28	0.97	541.41	7.41
50	408	75.16	1.29	456.18	7.57
75	674	74.38	1.10	413.27	4.35
100	959	71.15	2.00	383.56	8.55
200	2206	68.88	1.58	381.52	6.5
500	6468	69.04	0.89	371.11	6.76

COMMUNICATION OVERHEAD

- Transcipher ciphertext ~407 KiB, due to homomorphic encrypted symmetric key
- Bandwidth savings up to 50.4× at S=500
- Direct FHE only more efficient for smaller dataset sizes (S=10)

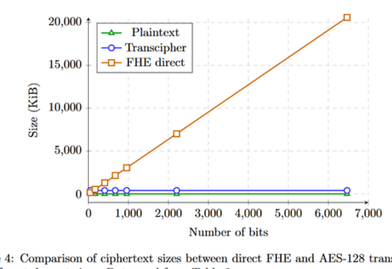

Figure 4: Comparison of ciphertext sizes between direct FHE and AES-128 transciphering for different dataset sizes. Data used from Table 2.

Table 2: Communication overhead of AES-128 transciphering per dataset size, showing plaintext size, direct FHE ciphertext size, transciphering ciphertext size, and bandwidth savings for using transciphering compared to direct FHE.

S	# Bits	Plaintext	FHE direct	Transcipher	Savings
10	48	6 B	152.81 KiB	407.52 KiB	0.38
25	168	21 B	534.84 KiB	407.53 KiB	1.31
50	408	51 B	1.27 MiB	407.56 KiB	3.20
75	674	85 B	2.10 MiB	407.59 KiB	5.28
100	959	120 B	2.98 MiB	407.62 KiB	7.49
200	2206	276 B	6.86 MiB	407.77 KiB	17.2
500	6468	809 B	20.11 MiB	408.30 KiB	50.4

AES-128 VS KREYVIUM

- AES-128 faster in total for S ≤ 25
- Kreyvium ~2.5× faster per-bit for all dataset sizes
- Kreyvium more efficient for larger datasets (up to 2.22× speedup)

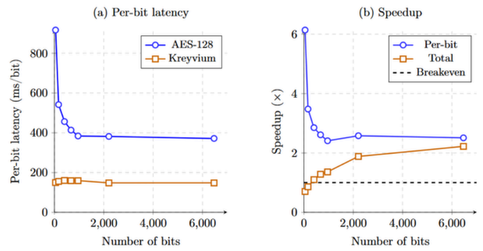

Figure 5: AES-128 vs Kreyvium: (a) per-bit latency and (b) speedup comparison. Data used from Table 3.

Table 3: AES-128 vs Kreyvium: computational overhead comparison per dataset size. The total time is calculated as Total = Init + (ms/bit × #bits)/1000. The total speedup is the factor saved using Kreyvium instead of AES-128.

		AES-128			Kreyvium			Speedup (×)	
S	# Bits	Init (s)	ms/bit	Total (s)	Init (s)	ms/bit	Total (s)	Per-bit	Total
10	48	74.44	916.17	118.42	161.72	149.22	168.88	6.14	0.70
25	168	74.28	541.41	165.24	167.12	155.55	193.25	3.48	0.86
50	408	75.16	456.18	261.28	171.24	159.92	236.49	2.85	1.10
75	674	74.38	413.27	352.92	169.91	158.57	276.79	2.61	1.28
100	959	71.15	383.56	438.98	170.45	159.09	323.02	2.41	1.36
200	2206	68.88	381.52	910.51	158.09	147.57	483.63	2.58	1.88
500	6468	69.04	371.11	2468.37	157.82	147.69	1113.08	2.51	2.22

CONCLUSION

- AES-128 transciphering becomes relatively more practical as datasets grow, but Kreyvium remains the faster choice
- Computational: fixed per-block cost spread over fewer data bits for smaller datasets → higher per-bit latency
- Communication: transciphering stays ~constant (~407KiB) vs linear growth for direct FHE
  - Up to 50.4x bandwidth saving
- AES-128 vs Kreyvium: Breakeven between S=25 and S=50
  - AES cheaper for smaller dataset sizes
  - Kreyvium up to 2.22x faster for larger sizes

FUTURE WORK

Future work that can be done based on this research:

- Implement the decryption circuit using Programmable Bootstrapping (PBS) via the Shortint API instead of the Boolean circuit
- Extend the comparison to include other HE-friendly ciphers such as HERA and Pasta
- Evaluate larger dataset sizes to further investigate scalability of AES-128 transciphering

Max Tanis, M.Tanis@student.tudelft.nl

CSE3000 Research Project

Supervisor: Zeki Erkin, TU Delft