RevoLUT : Rust Efficient and Versatile Oblivious Look-Up-Table

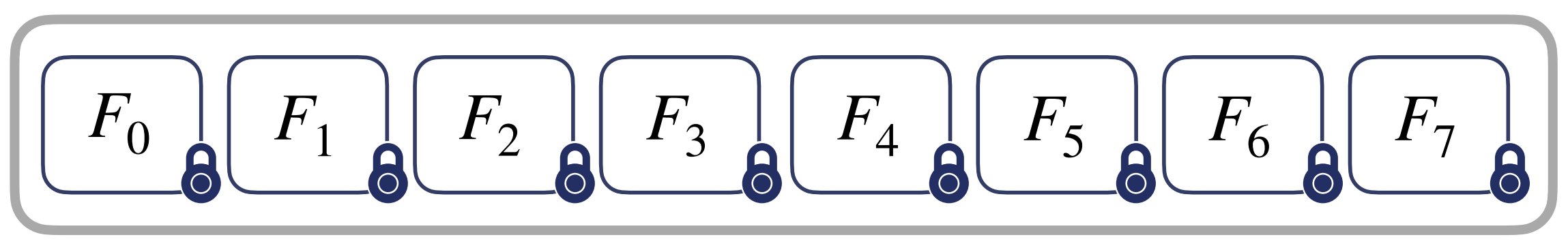
Sofiane Azogagh, Zelma Aubin Birba, Marc-Olivier Killijian, Sébastien Gambs, Félix Larose-Gervais

University of Quebec at Montreal, Canada

ABSTRACT

RevoLUT is a rust library built upon **tfhe-rs** that reimagines the use of Look-Up-Tables (LUT) beyond their conventional role in function encoding, as commonly used in TFHE's programmable boostrapping. Instead, RevoLUT leverages LUTs as first class objects, enabling efficient oblivious operations such as **array access**, **elements sorting** and **permutation** directly within the table. This approach supports oblivious algorithm, providing a secure, privacy-preserving solution for handling sensitive data in various applications.

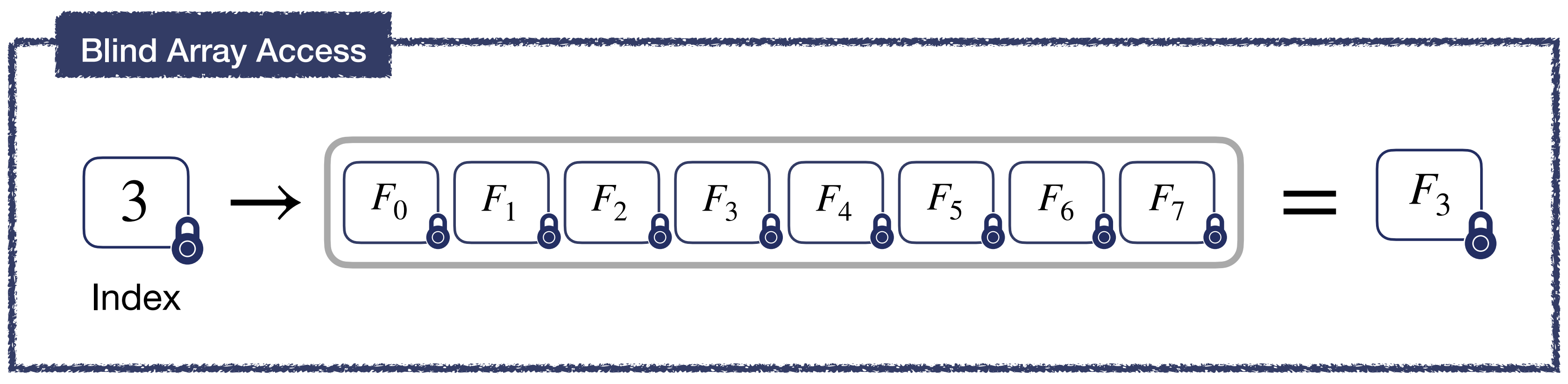
TFHE's LUT



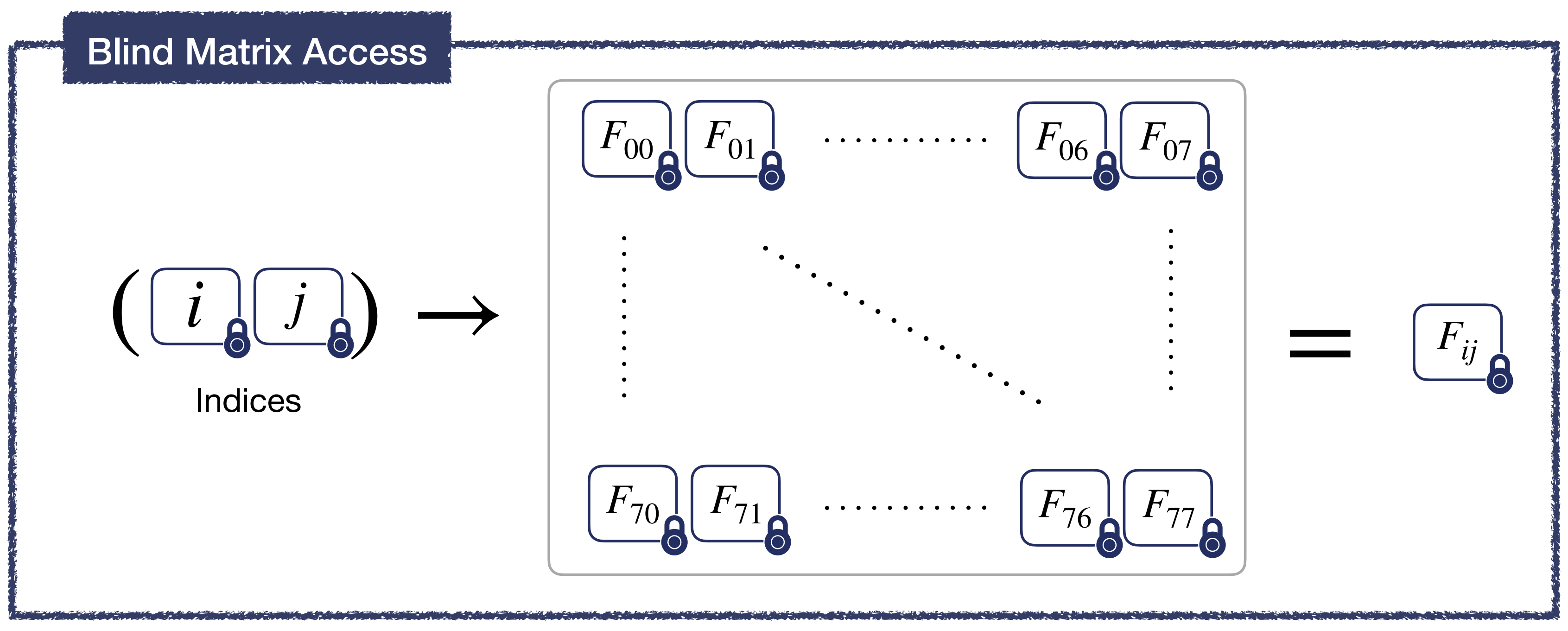
Private Look-Up-Table

Encrypted as an RLWE ciphertext with redundancy, we can consider a Private Look-Up-Table as a first class object to store private data.

BLIND READ

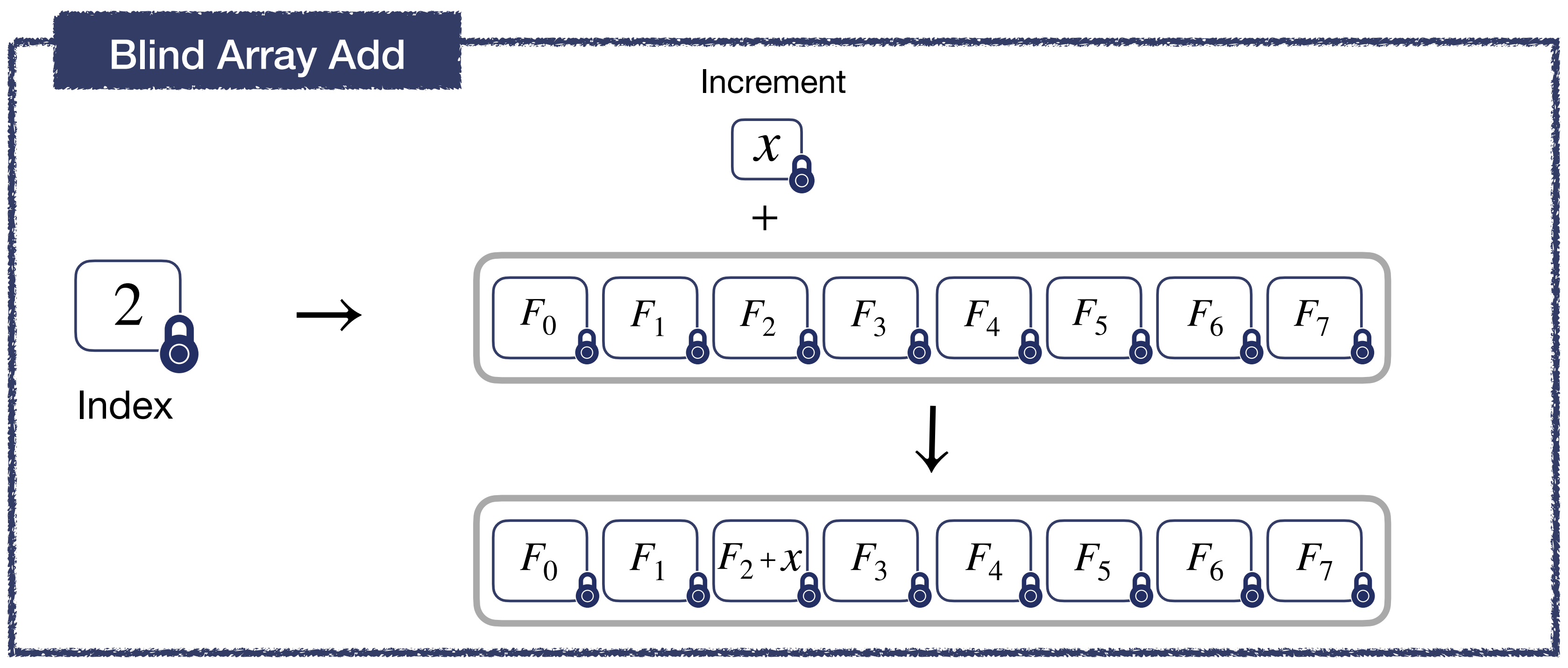

Blind Array Access

As such, the programmable bootstrapping implemented in tfhe-rs is then interpreted as a **Blind Array Access.**

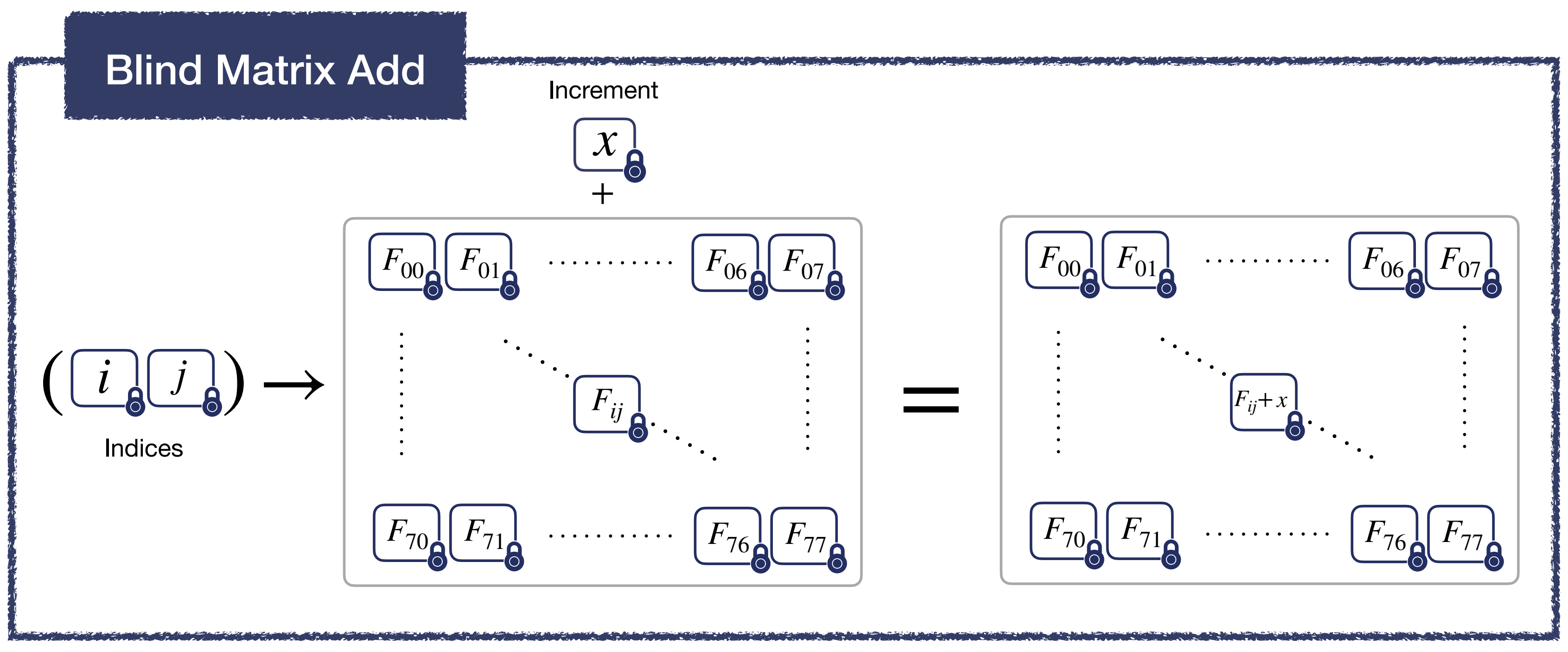

Blind Matrix Access
$(i\ j)$
Indices
F_{00} F_{01} F_{06} F_{07}
F_{70} F_{71} F_{76} F_{77}
=
F_{ij}

As an extension of the previous primitive, we developed **Blind Matrix Access**, a method tailored for private read on encrypted matrices.

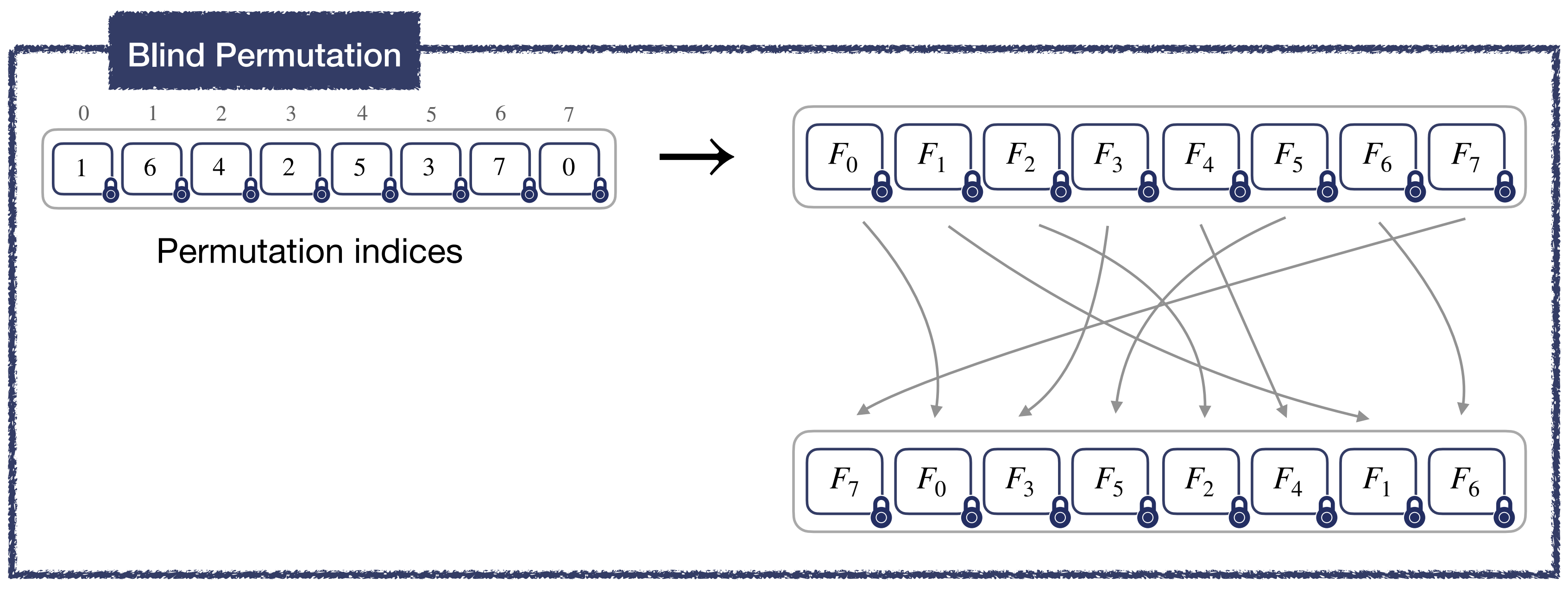
BLIND WRITE


Blind Array Add
Increment
x
+
2
Index
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7
F_0 F_1 F_2+x F_3 F_4 F_5 F_6 F_7

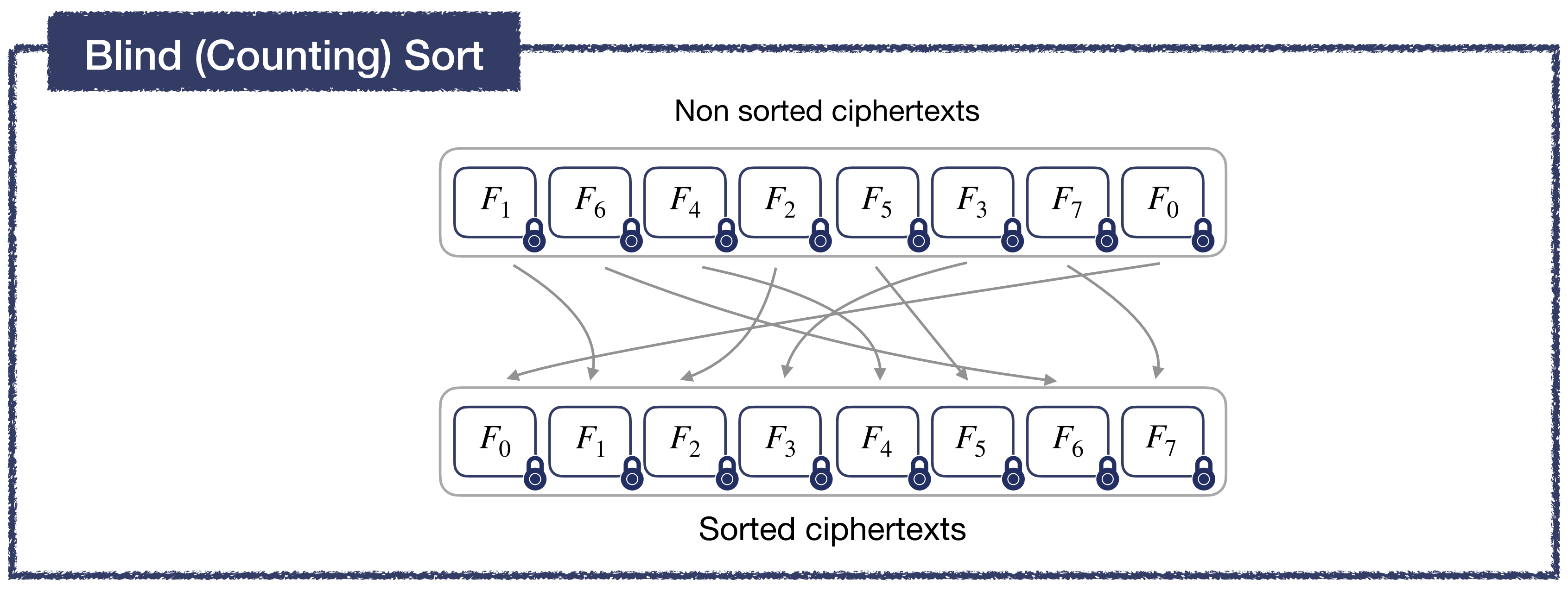
Using TFHE's Public Functional Key Switching and Blind Rotation, we can also write into the Look-Up-Tables or the encrypted matrices.


Blind Matrix Add
Increment
x
+
$(i\ j)$
Indices
F_{00} F_{01} F_{06} F_{07}
F_{ij}
F_{70} F_{71} F_{76} F_{77}
=
F_{00} F_{01} F_{06} F_{07}
$F_{ij}+x$
F_{70} F_{71} F_{76} F_{77}

BLIND ORDERING

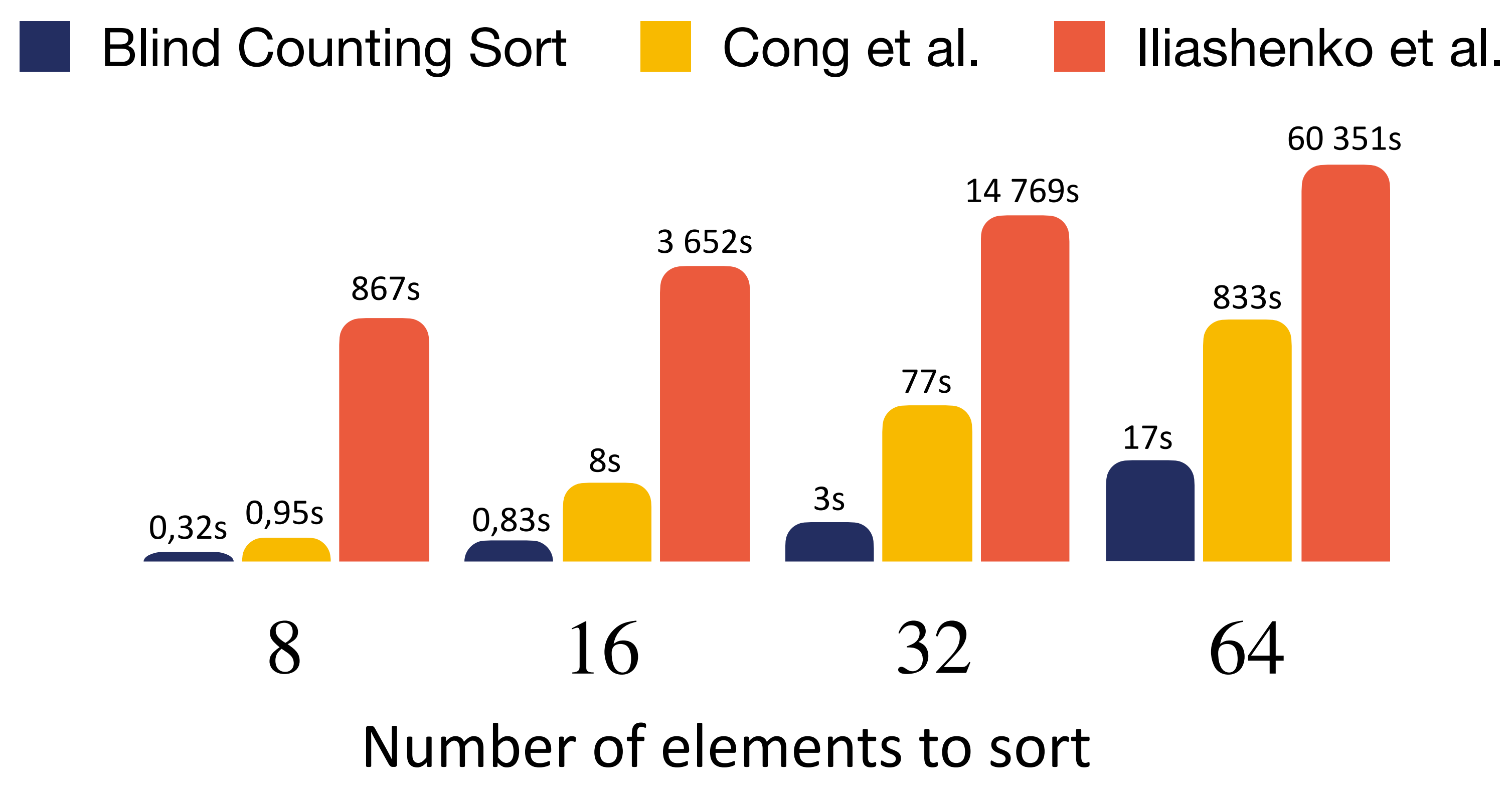

Blind Permutation
0 1 2 3 4 5 6 7
1 6 4 2 5 3 7 0
Permutation indices
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7
F_7 F_0 F_3 F_5 F_2 F_4 F_1 F_6

Multiple Blind Rotations can be applied to reorder the elements in the LUT according to an encrypted sequence of permutation indices.


Blind (Counting) Sort
Non sorted ciphertexts
F_1 F_6 F_4 F_2 F_5 F_3 F_7 F_0
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7
Sorted ciphertexts

Leveraging Blind Array Access and Blind Array Add enabled us to efficiently port the counting sort in FHE, making the first **non-comparison oblivious sort** in the FHE domain.This oblivious sort was then employed as a subroutine to develop a tournament-style **top-k selection**, which was subsequently integrated into an efficient private k-Nearest Neighbor algorithm in **Azogagh et al.**

PERFORMANCE GLIMPS



For more details including performances on the top-k selection and the private k-Nearest Neighbor, see **Azogagh et al.**

LINKS

sofianeazogagh/revoLUT

ia.cr/2024/1935

REFERENCES

Cong et al. : Revisiting Oblivious Top-k Selection with Applications to Secure k-NN Classification

Iliashenko et al. : Faster homomorphic comparison operations for BGV and BFV.

Azogagh et al. : A non comparison oblivious sort and its application to k-NN

UQÀM

DEEL
DEpendable & Explainable Learning