CRYPTO DARK MATTER ON THE TORUS: OBLIVIOUS PRFS FROM SHALLOW PRFS AND FHE

Martin R. Albrecht[1] Alex Davidson[2]
Amit Deo[3] Daniel Gardham[4]

[1]King's College London and SandboxAQ

[2]NOVA LINCS & DI, FCT, Universidade NOVA de Lisboa

[3]Crypto Quantique & Zama, [4]University of Surrey

FHE.org Meetup, 23rd November 2023

a.davidson@fct.unl.pt ::: eprint/2023/232

Post-quantum Partially Oblivious Pseudorandom Functions (POPRFs)

- Constructed from low-depth PRFs and TFHE
- Heuristic arguments for verifiability
- More efficient than previous PQ schemes
- Efficiency still behind classical

TLDR

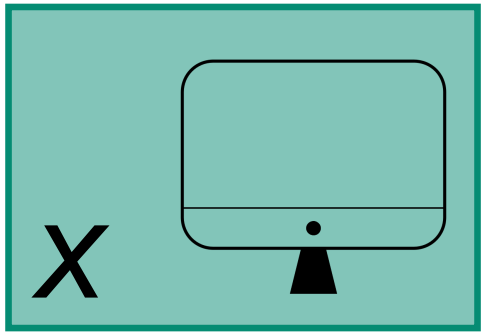
OPRFS AND APPLICATIONS

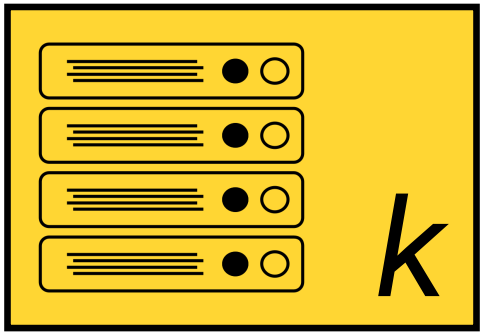
CORE CONSTRUCTION

OPRFS FROM LOW-DEPTH PRFS

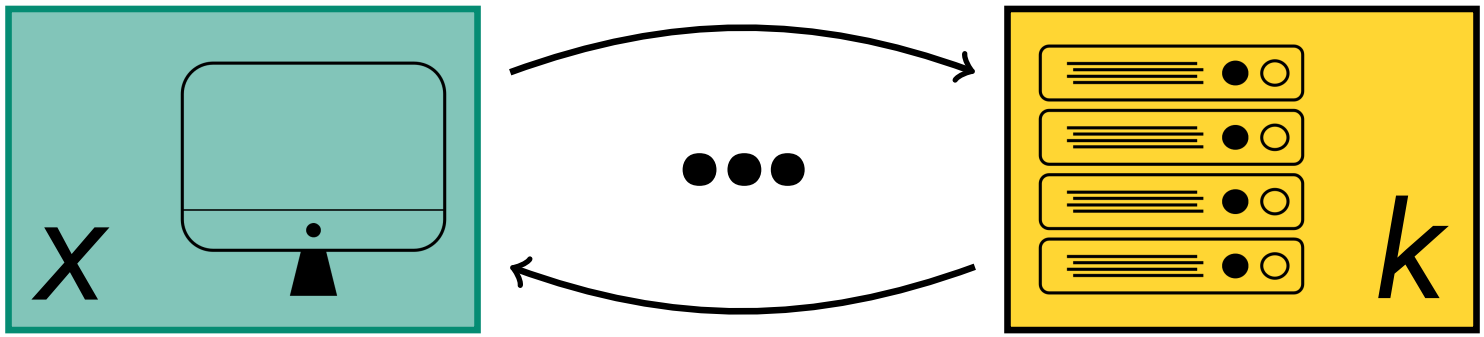
EFFICIENCY AND IMPLEMENTATION

CONCLUSIONS

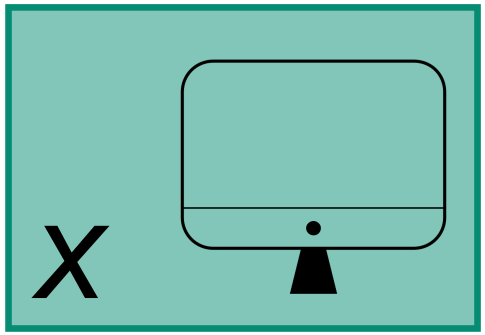


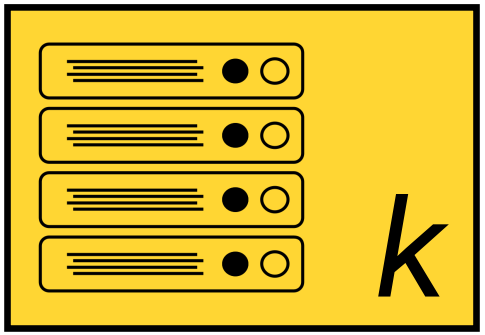


OBLIVIOUS PRFS: OPRFS



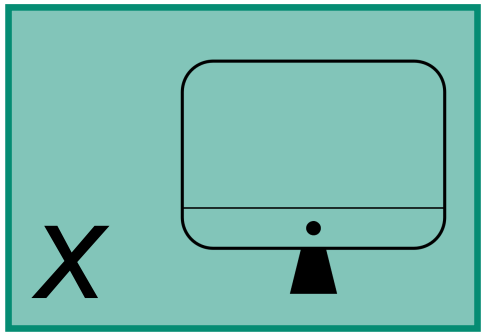
OBLIVIOUS PRFS: OPRFS

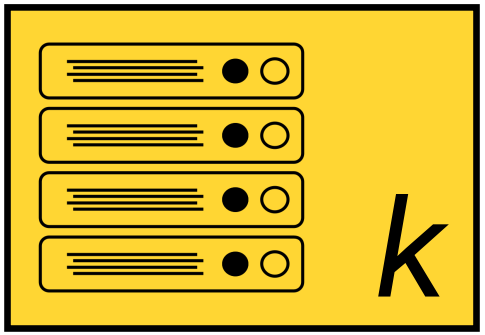




$\mathrm{PRF}(k, x)$

OBLIVIOUS PRFS: OPRFS

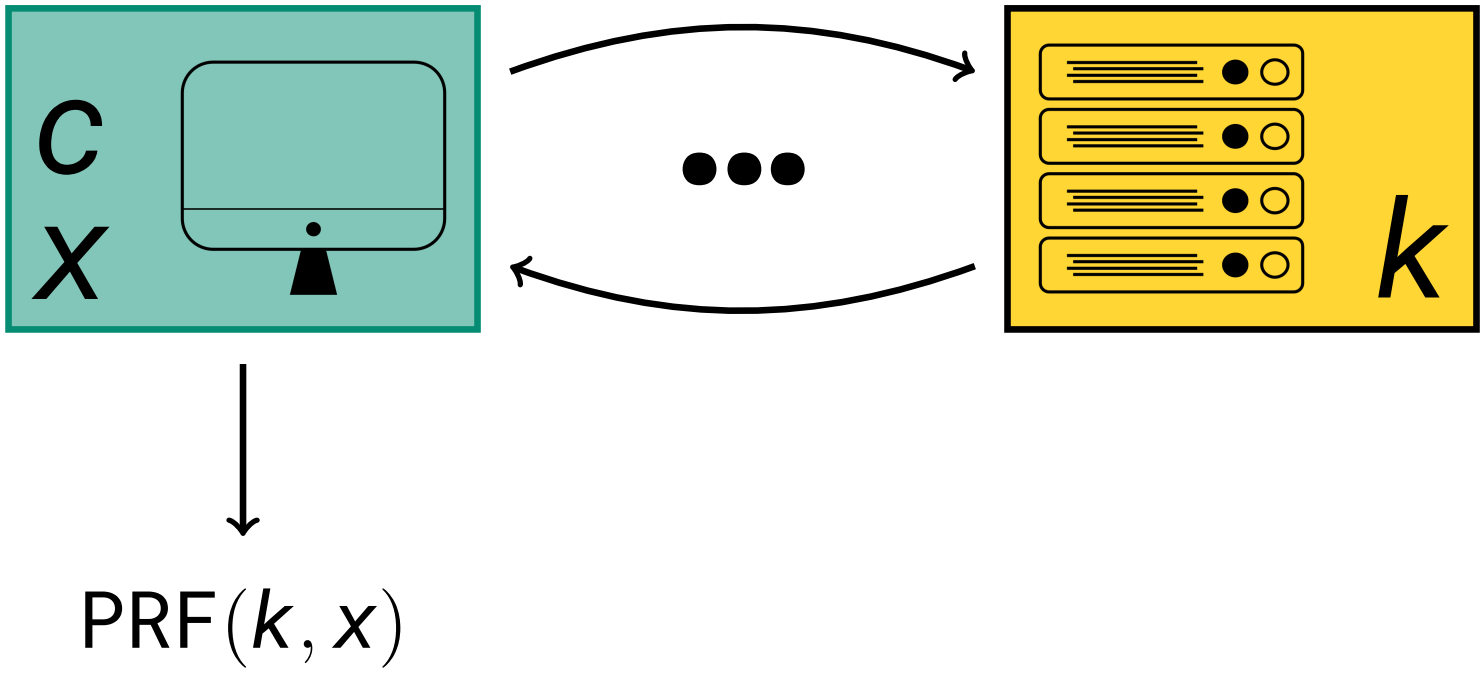




$$\mathrm{PRF}(k, x)$$

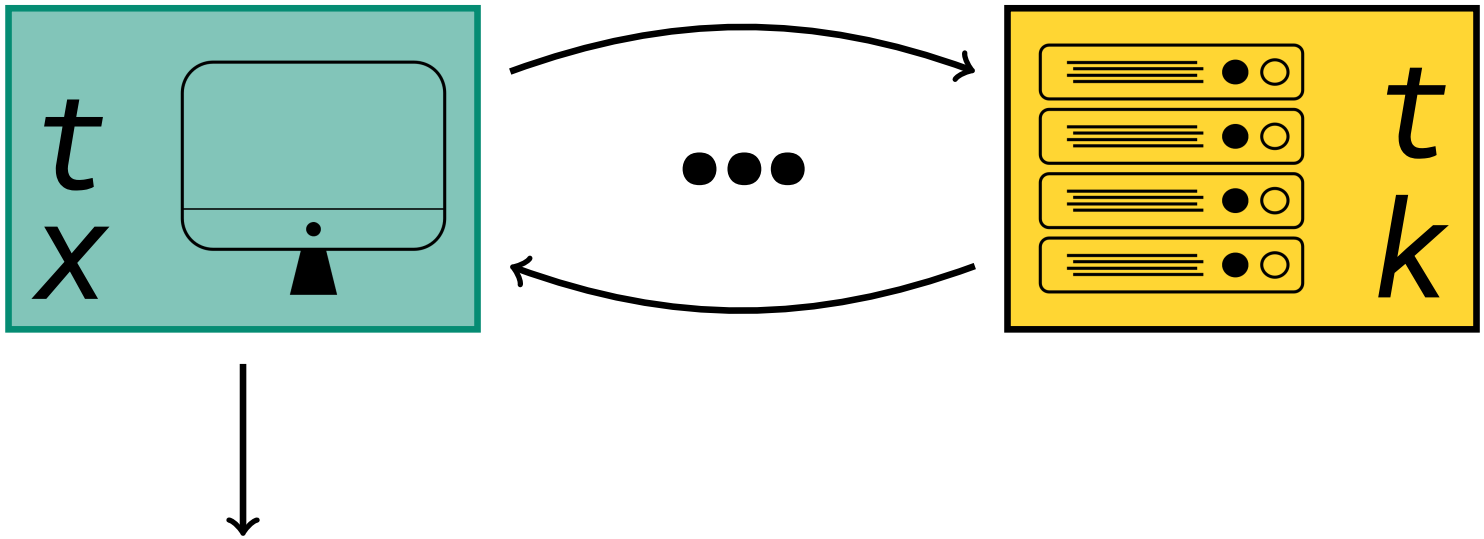
- Server learns nothing about x
- Client learns nothing about k

OBLIVIOUS PRFS: OPRFS



- Server proves that output was evaluated using k (malicious security)
- Requires an initial public commitment c
- Known as VOPRF

EXTRA PROPERTIES: VERIFIABILITY

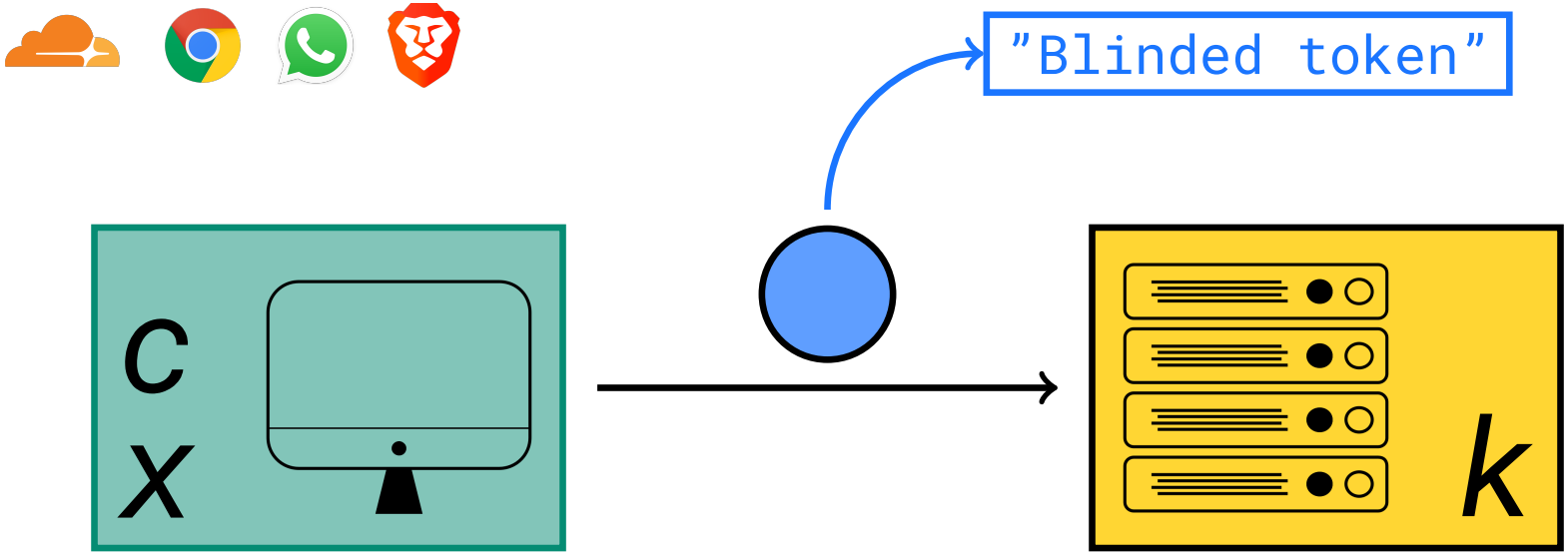


$\mathrm{PRF}(k, x; \mathtt{t})$

- Output $(\mathrm{PRF}(k, x; t), t)$ for public metadata t
- Allows adding arbitrary data into exchange
- Known as POPRF

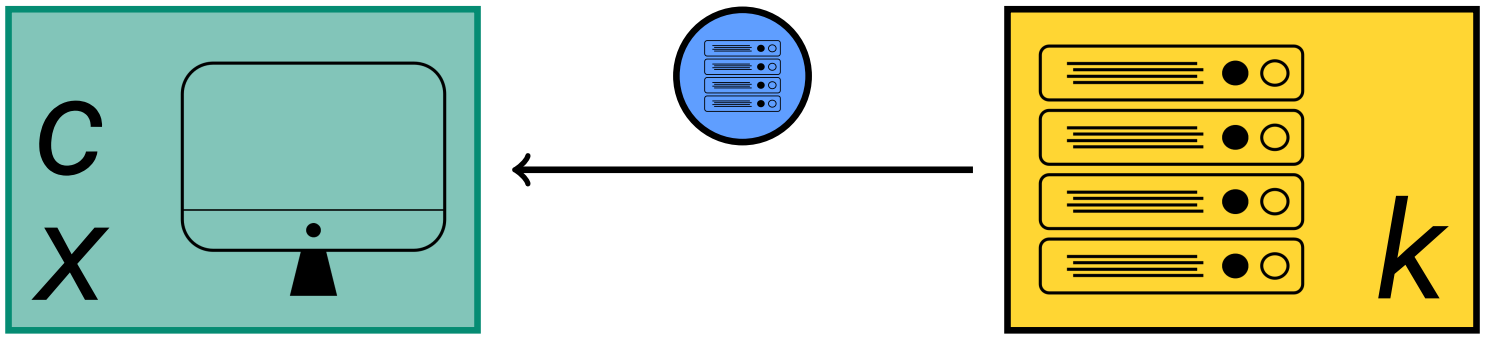
EXTRA PROPERTIES: PARTIALLY OBLIVIOUS

https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/



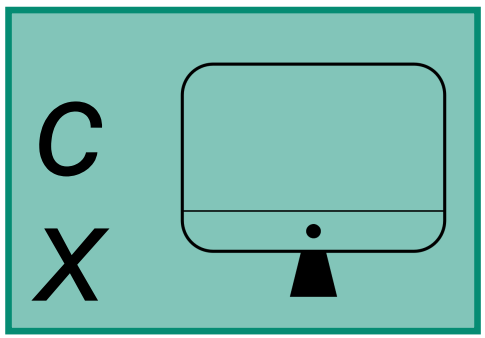
APPLICATIONS: PRIVACY PASS

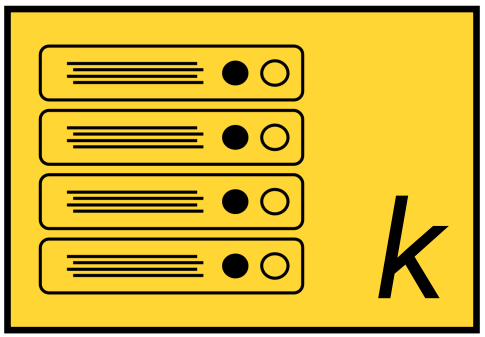
https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/

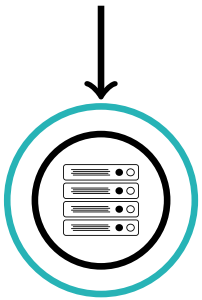


APPLICATIONS: PRIVACY PASS

https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/



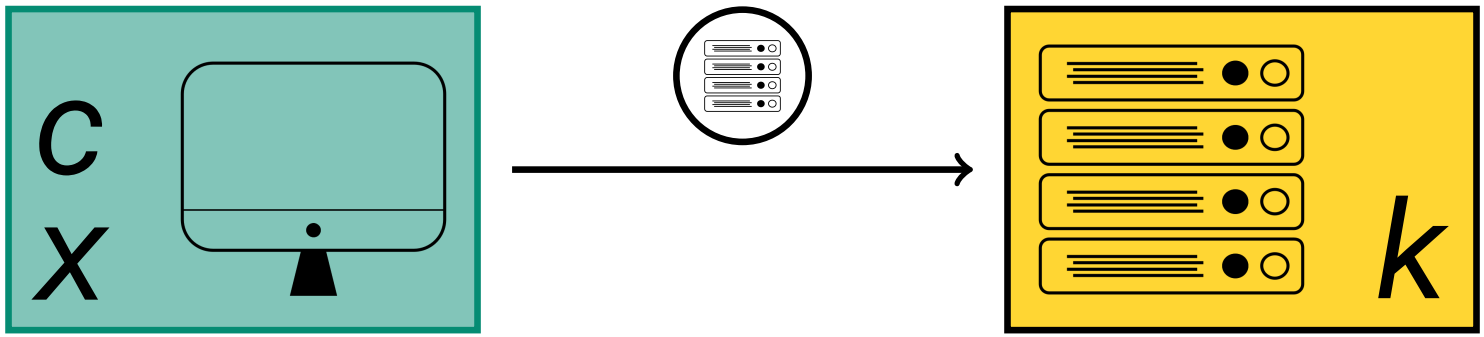




VOPRF output

APPLICATIONS: PRIVACY PASS

https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/



APPLICATIONS: PRIVACY PASS

- OPAQUE is a password-authenticated key exchange (PAKE) undergoing standardisation
- Allows mutual client-server authentication secure against pre-computation attacks
- Compiles an OPRF and authenticated key-exchange to build secure PAKEs

---

https://datatracker.ietf.org/doc/draft-irtf-cfrg-opaque/

APPLICATIONS: OPAQUE

OPRFS AND APPLICATIONS

CORE CONSTRUCTION

OPRFS FROM LOW-DEPTH PRFS

EFFICIENCY AND IMPLEMENTATION

CONCLUSIONS

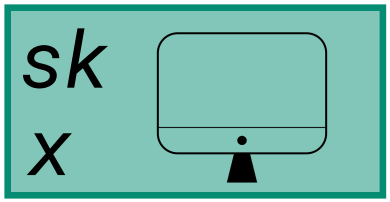
- (Ideal) lattices: Albrecht et al. (PKC21)

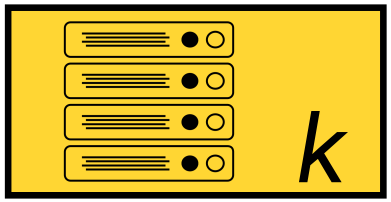
Poor performance compared with classical schemes

- Isogenies:
  - Boneh et al. (AC20)
  - Basso (SAC23)

Also poor performance & unsteady security guarantees

PREVIOUS APPROACHES

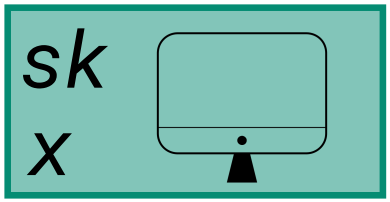


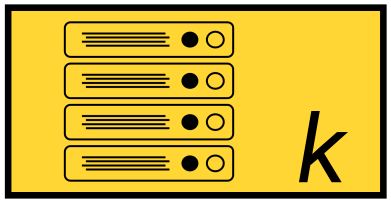


$$c_x = \mathsf{FHE.Enc}(sk, x)$$

Blinded by encryption

OUR IDEA

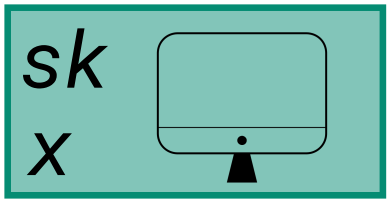


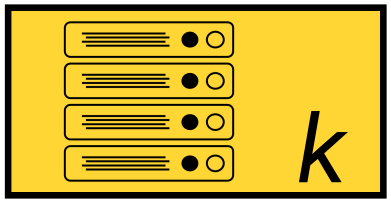


$$c_x = \mathsf{FHE.Enc}(sk, x) \longrightarrow$$

$$c_{k,x} = \mathsf{FHE.Enc}(sk, \mathsf{PRF}(k, x)) \longleftarrow$$

OUR IDEA



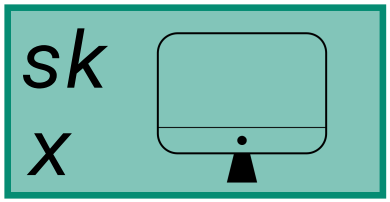


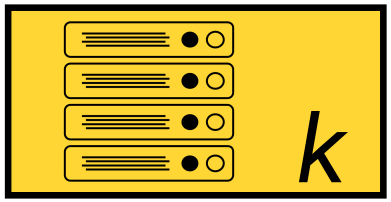
$$\mathsf{c}_x = \mathsf{FHE.Enc}(sk, x) \longrightarrow$$

$$\mathsf{c}_{k,x} = \mathsf{FHE.Enc}(sk, \mathsf{PRF}(k, x)) \longleftarrow$$

$$z_x = \mathsf{FHE.Dec}(sk, \mathsf{c}_{k,x}) = \mathsf{PRF}(k, x)$$

OUR IDEA





Proof of well-formed ciphertext?

$c_x = \mathsf{FHE.Enc}(sk, x)$

$\longrightarrow$

$c_{k,x} = \mathsf{FHE.Enc}(sk, \mathsf{PRF}(k, x))$

$\longleftarrow$

How expensive to evaluate PRF in HE?

$y_x = \mathsf{FHE.Dec}(sk, c_{k,x})$

$= \mathsf{PRF}(k, x)$

How to add verifiability?

EXTRA CONSIDERATIONS

OPRFS AND APPLICATIONS

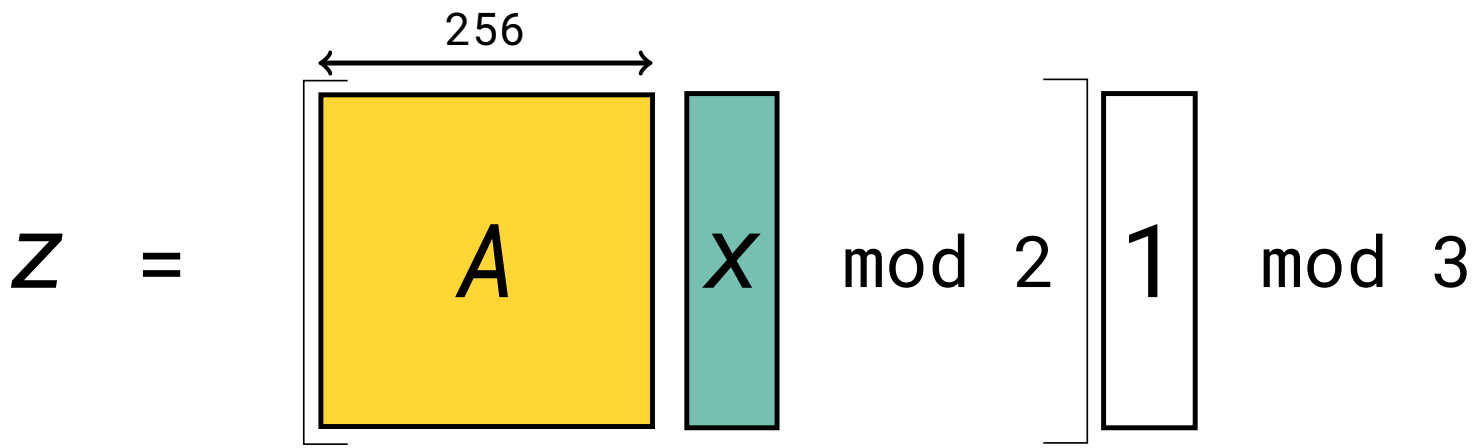
CORE CONSTRUCTION

OPRFS FROM LOW-DEPTH PRFS

EFFICIENCY AND IMPLEMENTATION

CONCLUSIONS

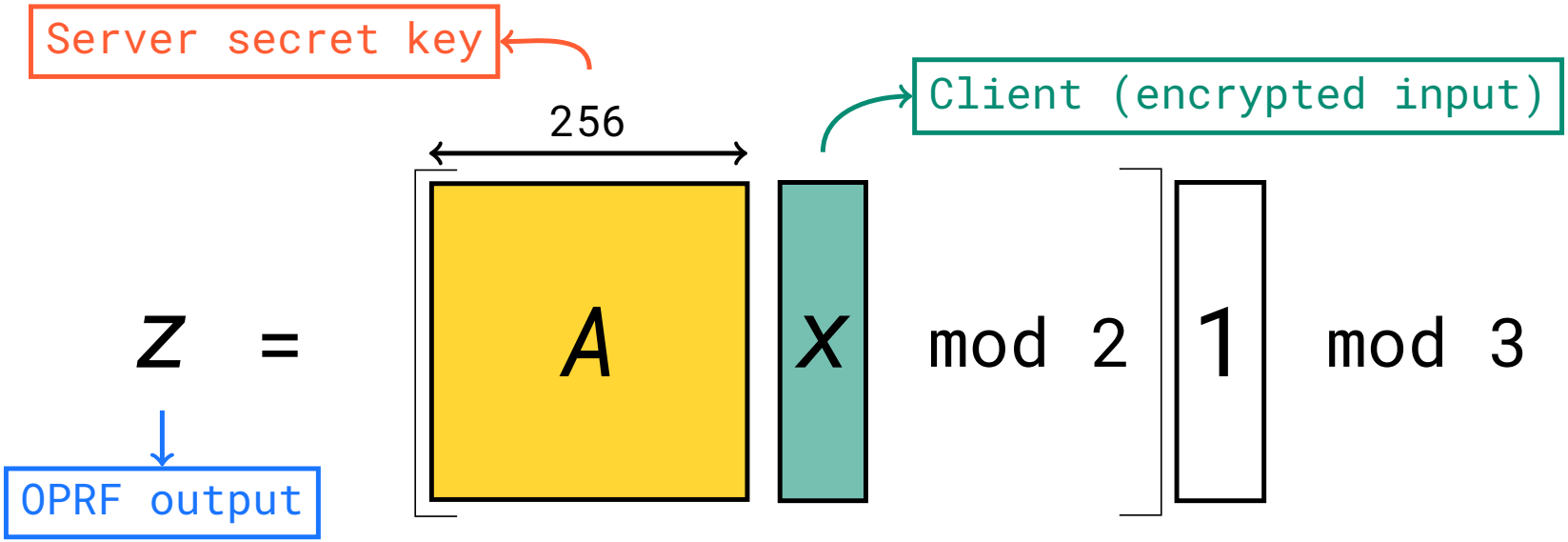
Boneh et al. low-depth (weak) PRF ($k = A$)



$A_{i,j} \in \{0,1\}, \quad x_i \in \{0,1\}$

Dan Boneh et al. “Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications”. In: *IACR Cryptol. ePrint Arch.* (2018), p. 1218. URL: https://eprint.iacr.org/2018/1218

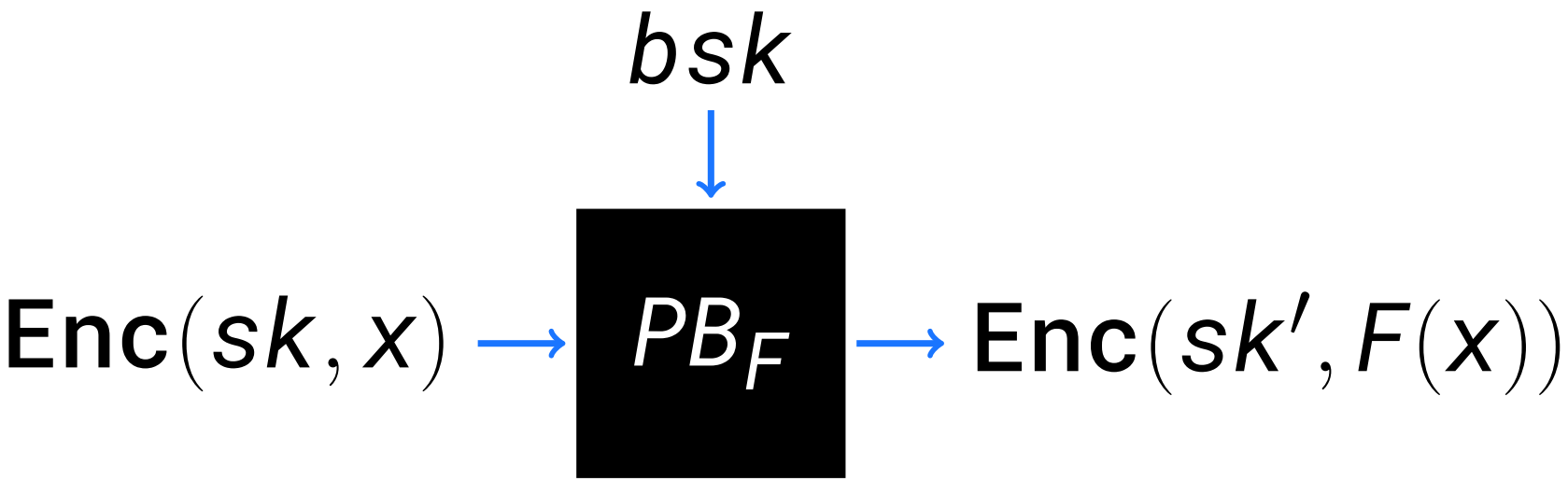
WEAK PRF



- 257 × (256 multiplications + 255 additions)
- Need to evaluate modular reductions in FHE

TFHE & Programmable Bootstrapping

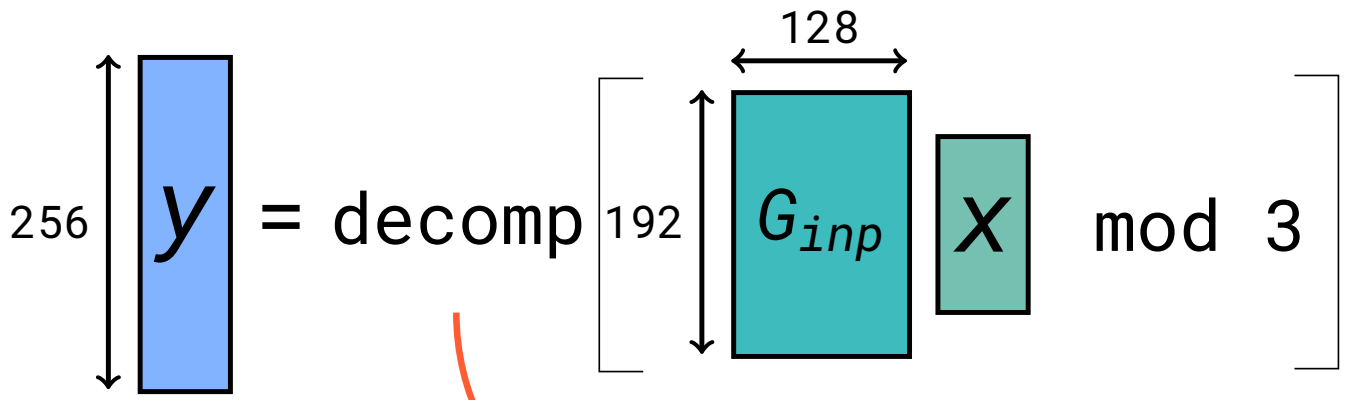
IDEA: WEAK OPRF IN TFHE



- TFHE supports lookup tables for F
- Implement modular reduction and switching

PROGRAMMABLE BOOTSTRAPPING: PBS

Boneh et al. low-depth (full) PRF

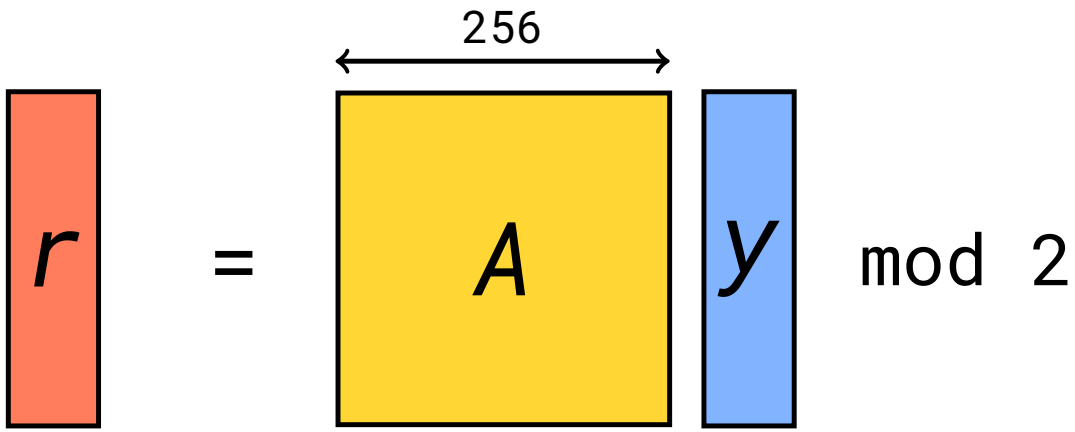


Maps 192 entries in {0,1,2} to 256 entries in {0,1}

- $G_{inp,i,j} \in \{0, \pm 1\}$, $x_i \in \{0, 1\}$
- Client-side (no encryption needed)

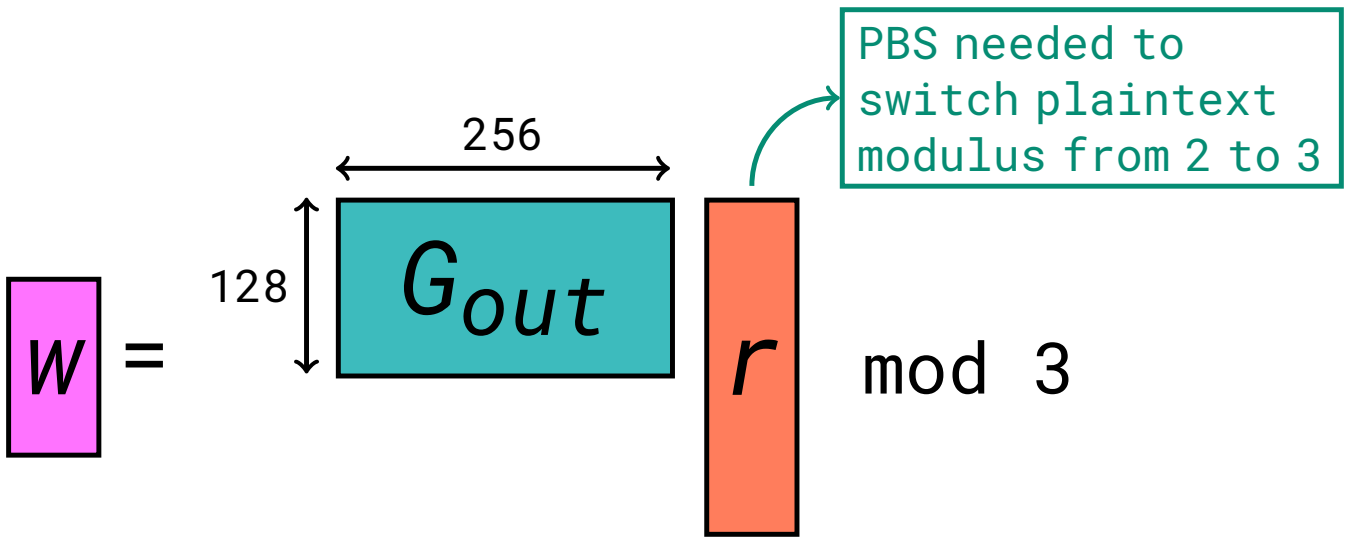
[Bon+18]

FULL PRF: STEP ONE



- $A_{i,j} \in \{0,1\}$, $x_i \in \{0,1\}$
- Server-side (no bootstrapping needed)

FULL PRF: STEP TWO



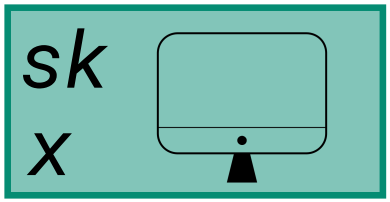
- $G_{out,i,j} \in \{0, \pm 1\}$, $x_i \in \{0, 1\}$
- Server-side

FULL PRF: STEP THREE

$$z = RO\left[\,w\,\right] \in \{0,1\}^{128}$$

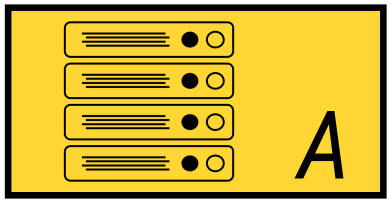
- Client-side (after decryption)

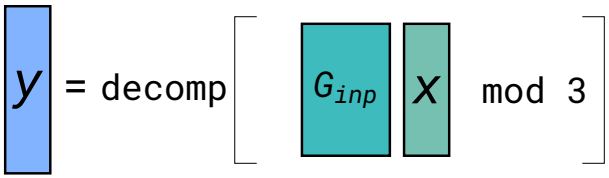
FULL PRF: STEP FOUR



bsk

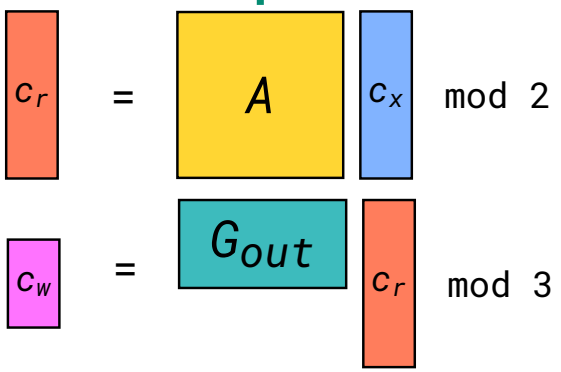
(offline)





$c_x = \mathsf{FHE.Enc}(sk, y)$

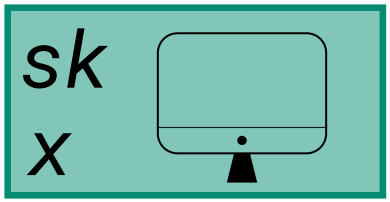
$A_t = RO(A, t \in \{0,1\}^*)$ for POPRF



c_w

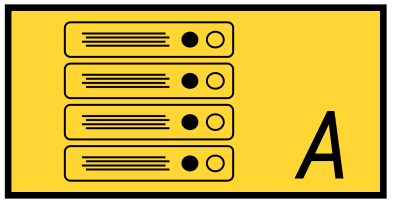
$z = RO(\mathsf{FHE.Dec}(sk, c_w))$

FULL PROTOCOL



(bsk, π_{bsk})

(offline)



π_{bsk} = Proof of correct bootstrapping key

$y = \text{decomp}\left[G_{inp} \; x \right] \bmod 3$

$(c_x = \text{FHE.Enc}(sk, y), \pi_{x,y})$

$\pi_{x,y}$ = Proof of well-formed ciphertext

$c_r = A \; c_x \bmod 2$

$c_w = G_{out} \; c_r \bmod 3$

c_w π_A

π_A = Proof of correct evaluation

$z = RO(\text{FHE.Dec}(sk, c_w))$

ZK PROOFS

- Server key is encoded in error of (RLWE) ciphertext
- Need mechanism for sanitizing ciphertexts
- Options:
  - Noise flooding
  - Additional bootstrapping **[Ec:DucSte16]**
  - TFHE-based randomised blind rotations **[EPRINT:Klu22]**

Part of bootstrapping procedure

CIRCUIT-PRIVATE BOOTSTRAPPING

- Bootstrapping key (π_{bsk}): [Crypto:LNP22], LaBRADOR
- Well-formed ciphertexts ($\pi_{x,y}$): [Crypto:LNP22], LaBRADOR
- Verifiability (π_A): *novel heuristics*

Can be combined into single proof

Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General". In: *CRYPTO 2022*, 2022. URL: https://doi.org/10.1007/978-3-031-15979-4%5C_3

ZKP INSTANTIATIONS

- Server publishes (verifiably) multiple checkpoints
  - $(x_i^*, z_i^* = PRF(sk, x_i^*))_{i \in [1^\lambda]}$.
- Client batched checkpoints with standard queries and shuffles:
  - $(x_5^*, x_1, x_{10}, \ldots, x_{22}^*, x_2)$
- Server evaluates OPRF on each input
- Client checks $z_j^* = PRF(sk, x_j^*)$ for each known answer test

VERIFIABILITY

Publish enough checkpoints

Options for malicious server:

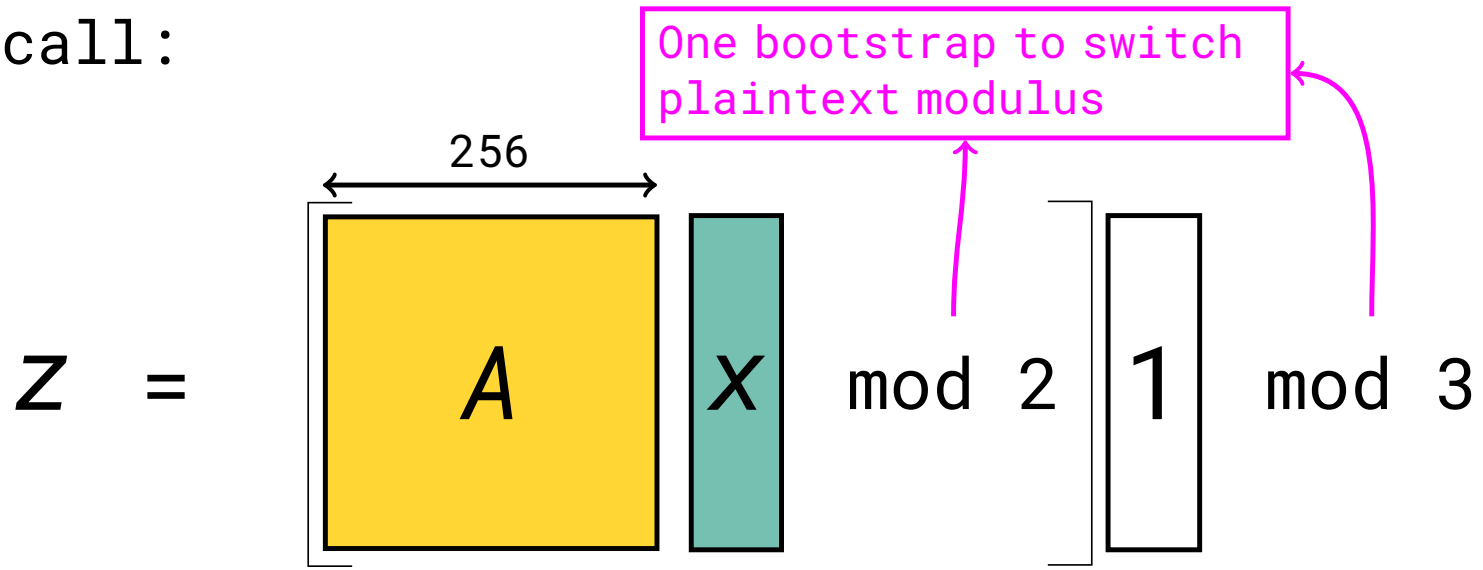
1. Must guess positions of checkpoints

2. Homomorphically evaluate cheating circuit C^*
   - C^* is correct on x_i^* for all i
   - Cheats on non-checkpoints

Limit *(heuristically)* homormorphic capability of server

MALICIOUS SERVER

Recall:



- Limit *bsk* to one homomorphic level (i.e. one bootstrap)
- **Challenge**: find depth-one cheating circuit

HEURISTIC VERIFIABILITY

OPRFS AND APPLICATIONS

CORE CONSTRUCTION

OPRFS FROM LOW-DEPTH PRFS

EFFICIENCY AND IMPLEMENTATION

CONCLUSIONS

(Only considering round-optimal)

work	assumption	communication cost
PKC:ADDS21+	R(LWE) & SIS	≈ 2MB
EPRINT:Basso23	SIDH	3.0MB
EPRINT:HeiMeiRec23	CSIDH	21KB
Our OPRF (amortised, ×64)	lattices, **TCC:BIPSW18+**	≈14.8MB (one-time) + ≈5KB (online)
Our OPRF (pk comp./amortised)	lattices, **TCC:BIPSW18+**	≈2.5MB (one-time) + ≈10KB (online)

ASYMPTOTIC PARAMETERS (OPRF)

(Only considering round-optimal)

work	assumption	communication cost
PKC:ADDS21+	R(LWE) & SIS	$>$ 128GB
EPRINT:Basso23	SIDH	8.7MB
Our VOPRF (amortised, $\times 64$)	heuristic	$\approx$15MB (one-time) + 16.2 $\times$ 5.3KB (online)
Our VOPRF (pk comp./amortised)	heuristic	$\approx$2.8MB (one-time) + 16.2 $\times$ 10.0KB (online)
AC:JKK14 (2HashDH)	One-more gap-DH	$\approx$ 32B (one-time) $\approx$ 96B (online)

ASYMPTOTIC PARAMETERS (VOPRF)

Experiments with 100-bits of security:

- $q = 2^{32}$, $e = 900$, $d = 1024$, $\sigma = 4.0$

Implementation in tfhe-rs[1][2]:

- Client KeyGen: 1s
- Client Request: 28.9ms
- Server Evaluation: 151ms
- Client Finalize: 0.2ms

[1] Code: https://anonymous.4open.science/r/oprf-fhe-D2E7

[2] **64** Intel Xeon Gold 6252 CPU @ 2.10GHz cores (server-side) and **768GB** of RAM

TFHE IMPLEMENTATION

Disclaimer: tfhe-rs currently does not support various core functionality.

- Modified test polynomials for blind rotation in TFHE
- Non-power-of-two moduli
- Circuit privacy
- Public-key compression

Complete implementation (minus ZKPs) in sage.[3]

[3] see paper

TFHE IMPLEMENTATION

OPRFS AND APPLICATIONS

CORE CONSTRUCTION

OPRFS FROM LOW-DEPTH PRFS

EFFICIENCY AND IMPLEMENTATION

CONCLUSIONS

- PQ (VP)OPRFs from FHE
- Close to practical costs for core FHE PRF
  - $<$ 1s and O(KB) in online phase (amortised)
- ZKPs still seem restrictive
- Heuristic security for verifiability
- **Future work**: Can direct constructions be made more efficient?

CONCLUSIONS

Thanks for listening!

a.davidson@fct.unl.pt ::: eprint/2023/232