ROUND-OPTIMAL VERIFIABLE OBLIVIOUS PSEUDORANDOM FUNCTIONS FROM IDEAL LATTICES

Martin R. Albrecht[1] Alex Davidson[2]
Amit Deo[1,3] Nigel P. Smart[4]

[1]Information Security Group, Royal Holloway, University of London, UK

[2]LIP, Lisboa, Portugal & Cloudflare Portugal

[3]ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France

[4]COSIC-imec, KU Leuven, Belgium & Dept Computer Science, University of Bristol, UK

IACR PKC 2021 ::: 2021-05-11

::: Post-quantum Verifiable Oblivious Pseudorandom Function (VOPRF) protocol from lattice-based hardness assumptions

::: Security holds in the Quantum Random Oracle Model (QROM)

::: ***Disclaimer***: This is a ***feasibility*** result

- More **efficient** construction needed

CONTRIBUTIONS

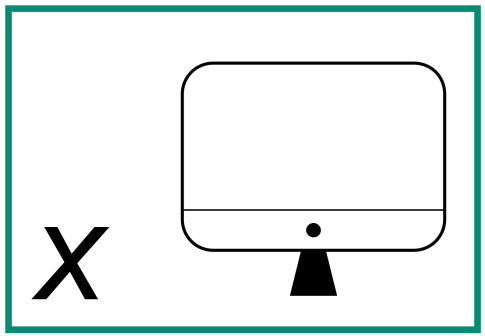
INTRODUCTION TO VOPRFS AND APPLICATIONS

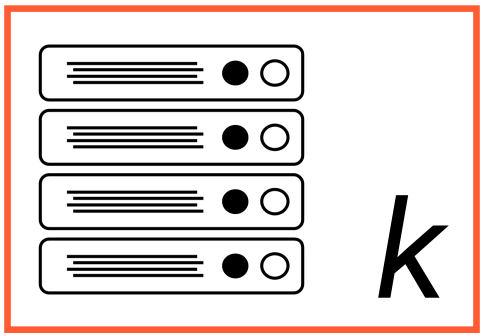
POST-QUANTUM CONSIDERATIONS

OUR VOPRF CONSTRUCTION

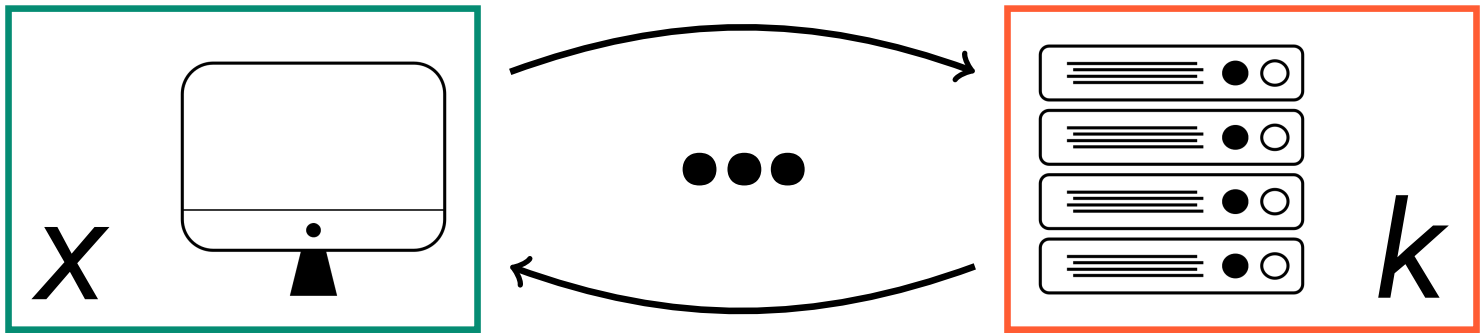
EFFICIENCY AND PARAMETERS

CONCLUSIONS AND OPEN PROBLEMS

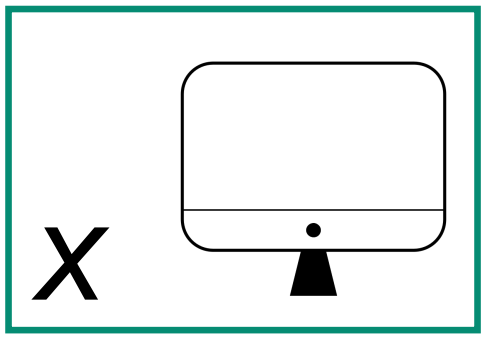


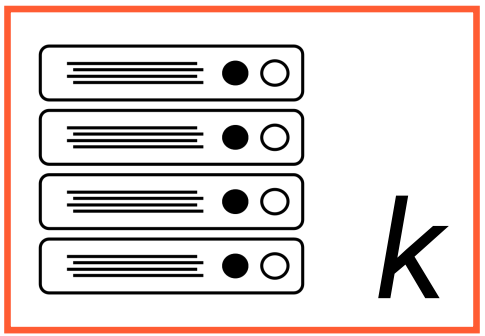


OBLIVIOUS PSEUDORANDOM FUNCTIONS



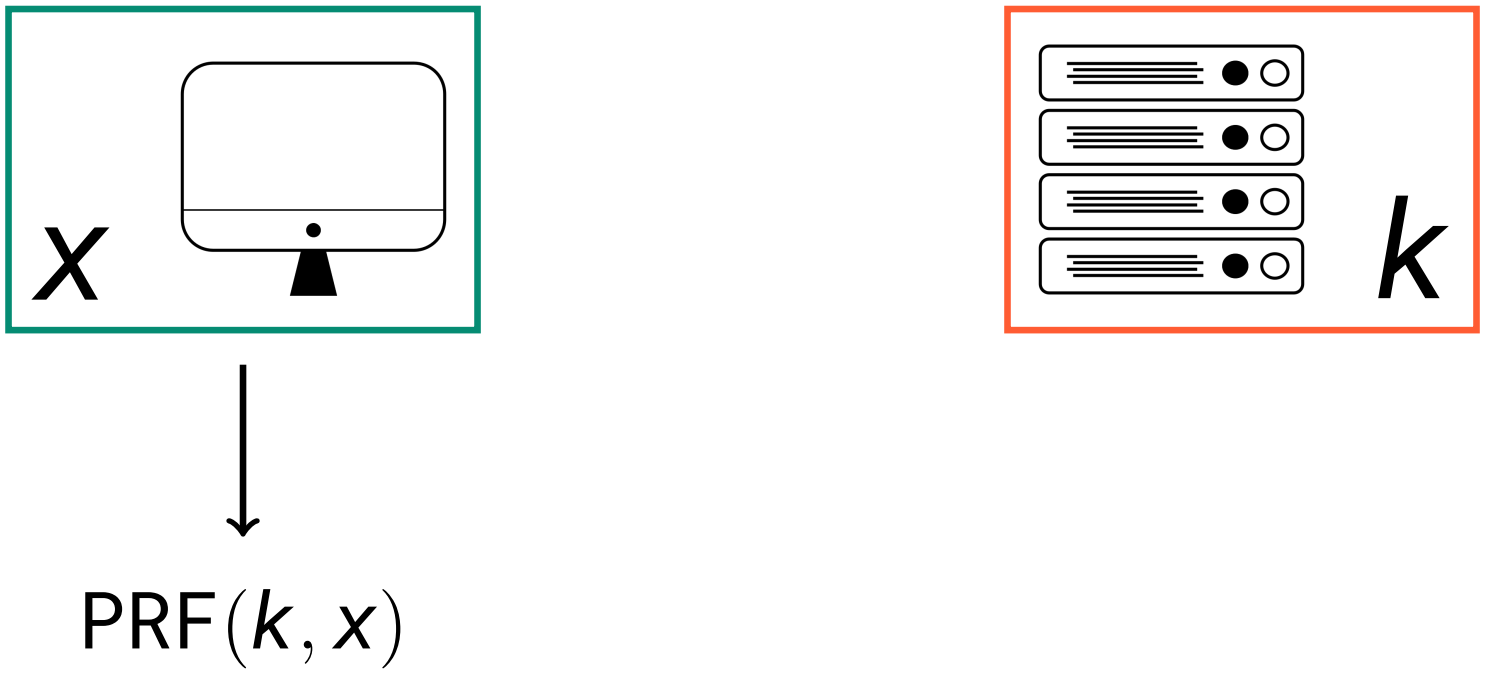
OBLIVIOUS PSEUDORANDOM FUNCTIONS





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

OBLIVIOUS PSEUDORANDOM FUNCTIONS

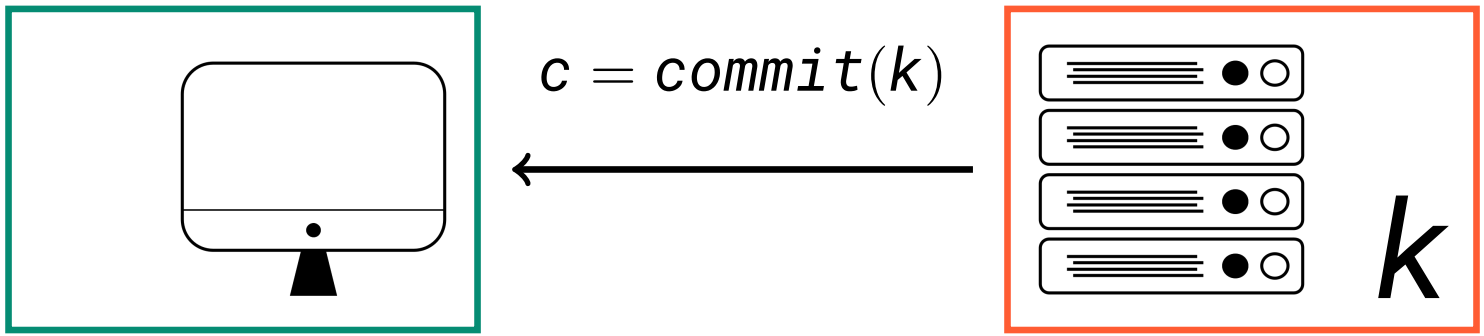


::: Server learns nothing about x

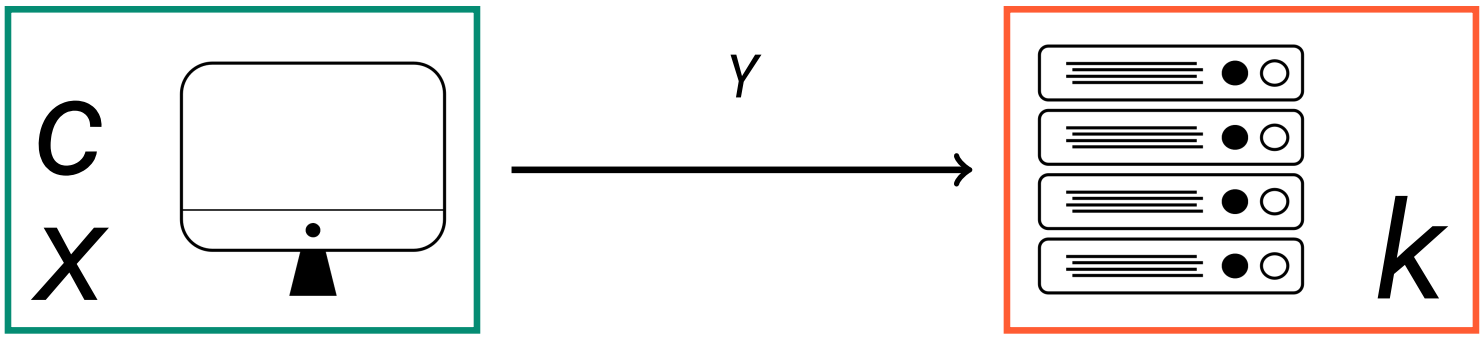
::: Client learns nothing about k

::: **Verifiability**: Server proves that output was evaluated using k

OBLIVIOUS PSEUDORANDOM FUNCTIONS



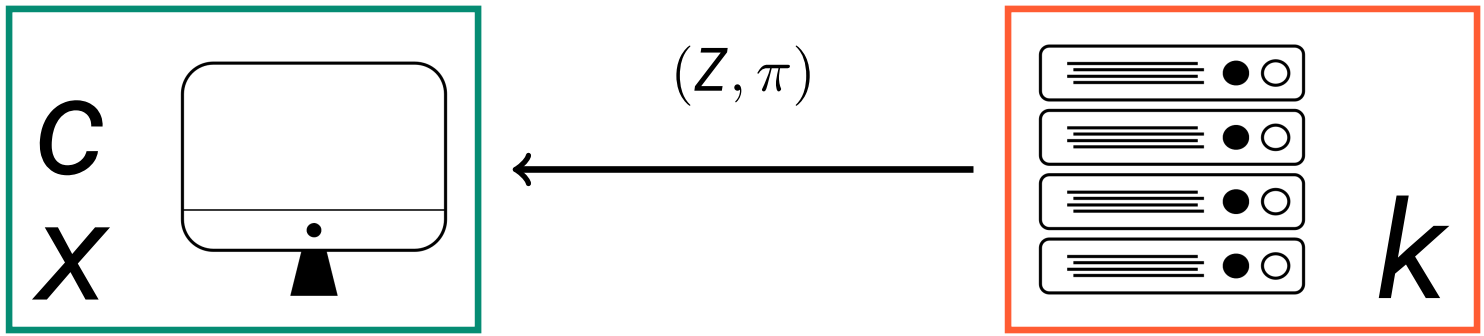
ONE-ROUND VOPRF: OFFLINE



$X = \text{encode}(x)$

$Y = blind(X)$

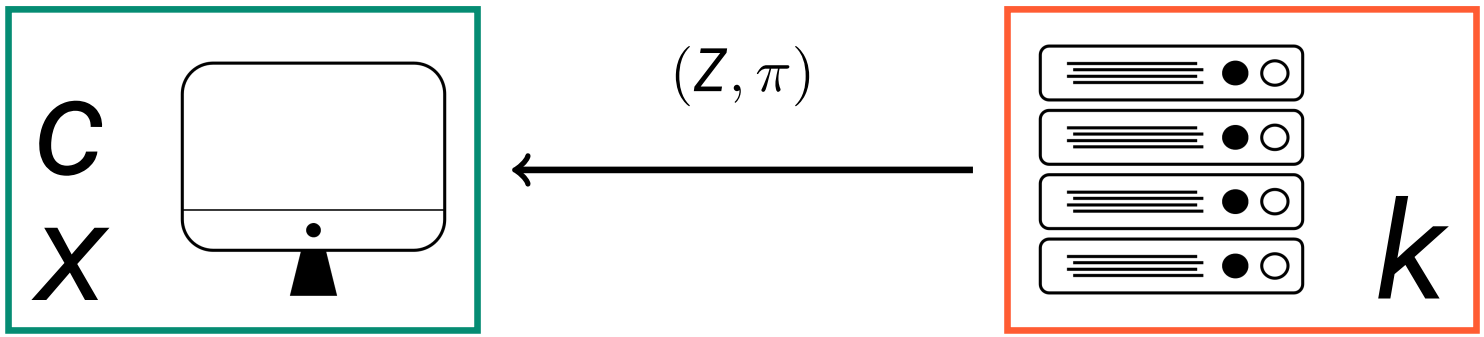
ONE-ROUND VOPRF: ONLINE



$Z = eval(k, Y)$

$\pi = \mathbb{P}(k; c, Y, Z)$

ONE-ROUND VOPRF: ONLINE



$1 =? \mathbb{V}(\pi, c, Y, Z)$

return *unblind*(Z)

$Z = eval(k, Y)$

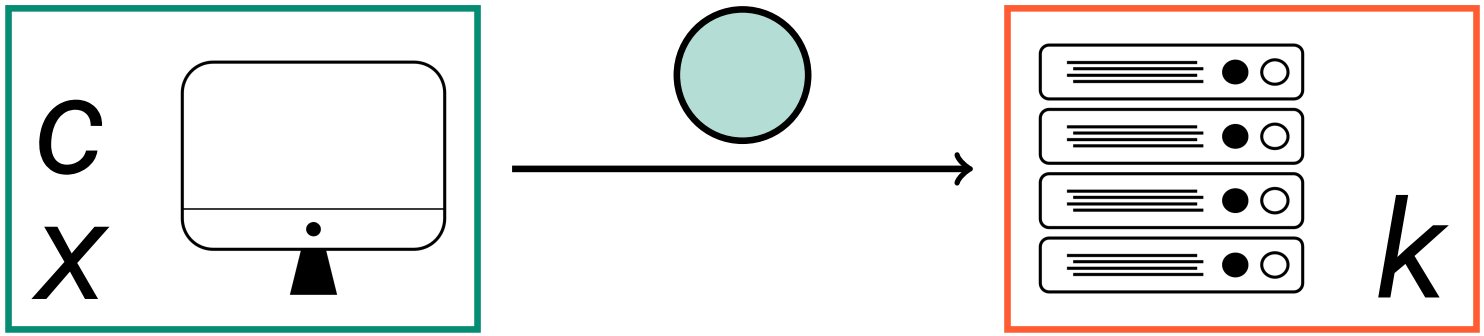
$\pi = \mathbb{P}(k; c, Y, Z)$

ONE-ROUND VOPRF: ONLINE

::: **Correctness**: $\mathit{unblind}(Z) = \mathsf{PRF}(k, \mathsf{encode}(x))$

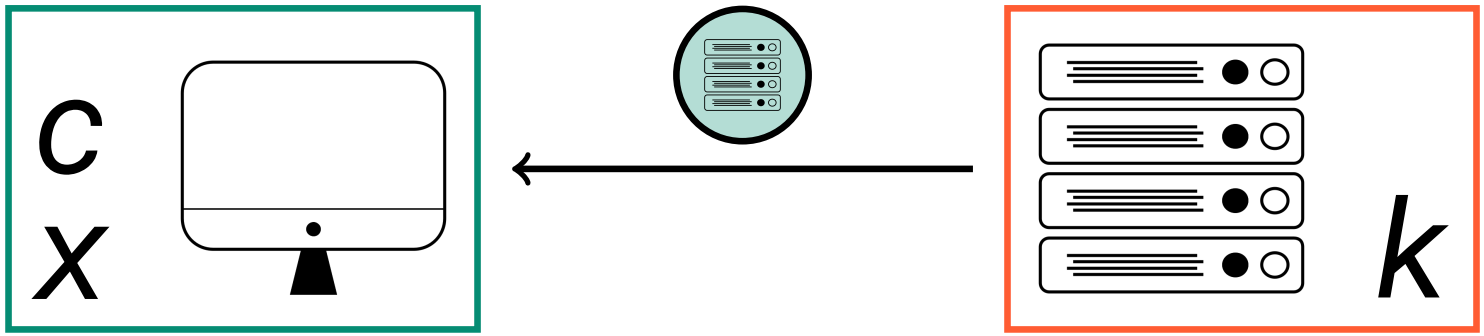
::: **Malicious security**: For any malicious client/server, there is a PPT simulator interacting with an ideal VOPRF functionality that can emulate their capabilities in the real system

REQUIREMENTS



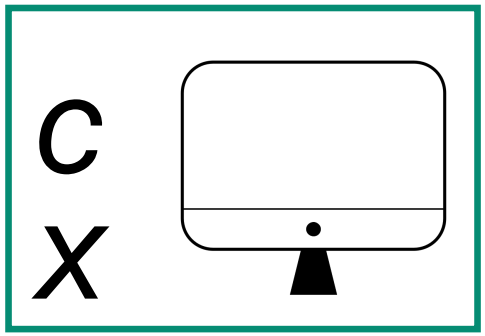
https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/

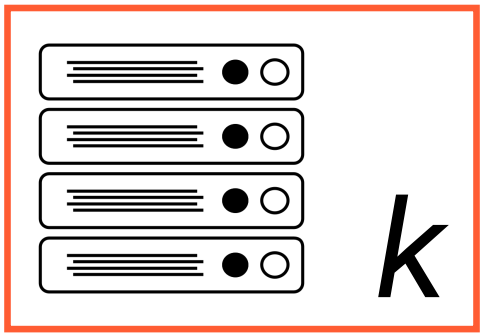
APPLICATIONS: PRIVACY PASS

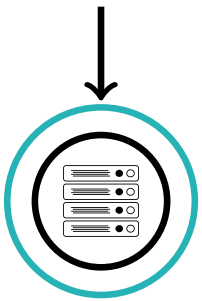


https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/

APPLICATIONS: PRIVACY PASS





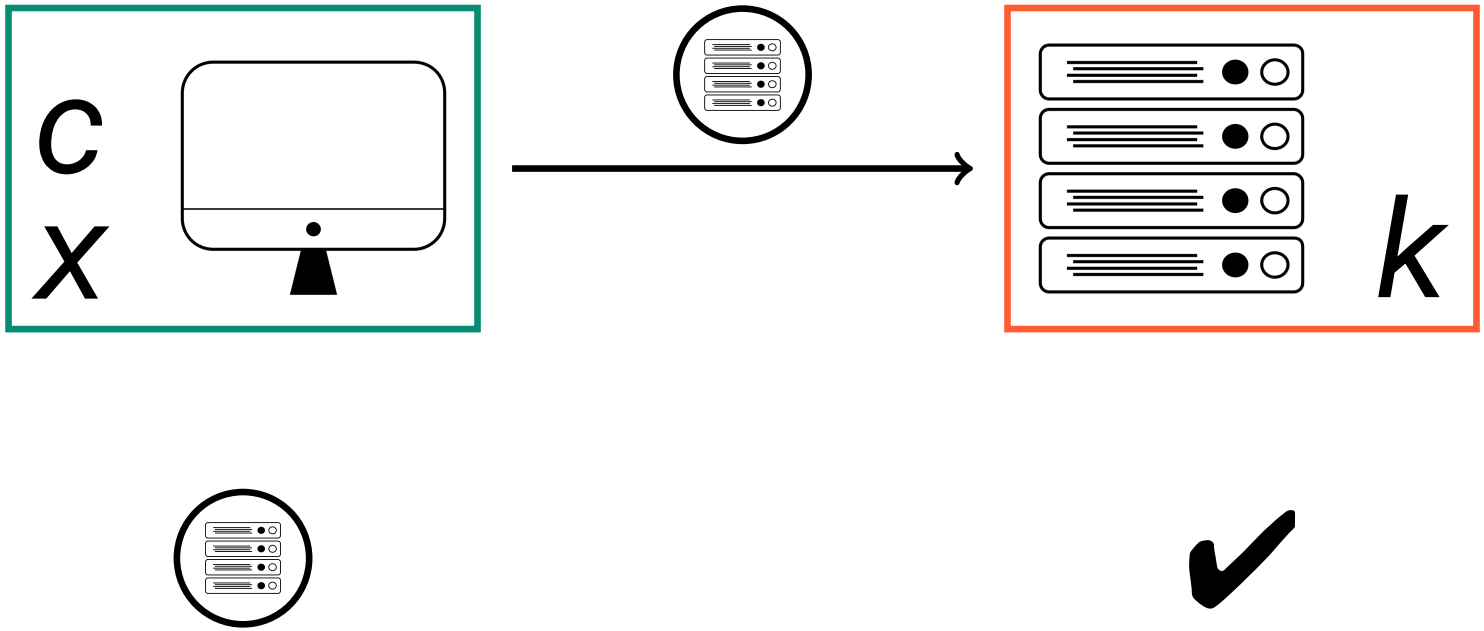


VOPRF
output

---

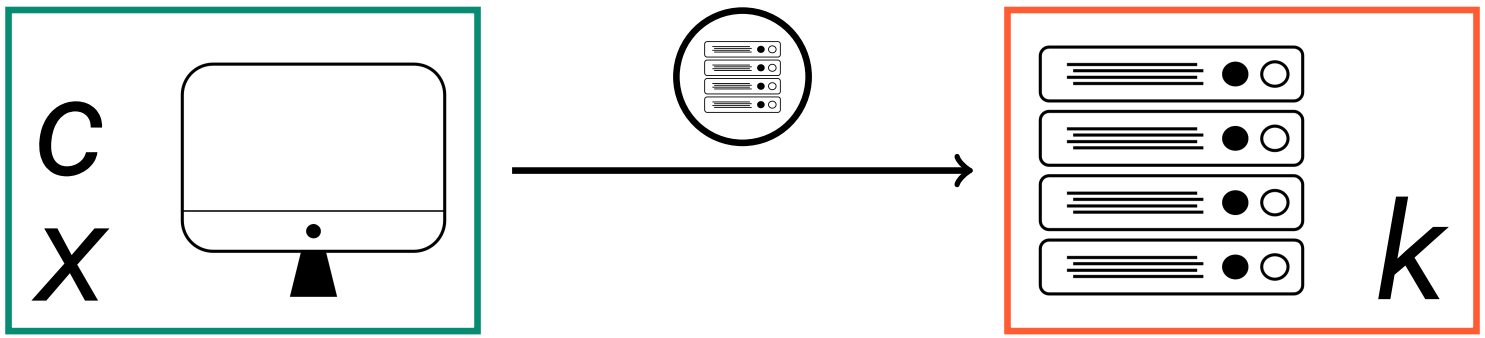
https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/

APPLICATIONS: PRIVACY PASS



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

APPLICATIONS: PRIVACY PASS



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

APPLICATIONS: PRIVACY PASS

::: OPAQUE is a password-authenticated key exchange (PAKE) undergoing standardisation

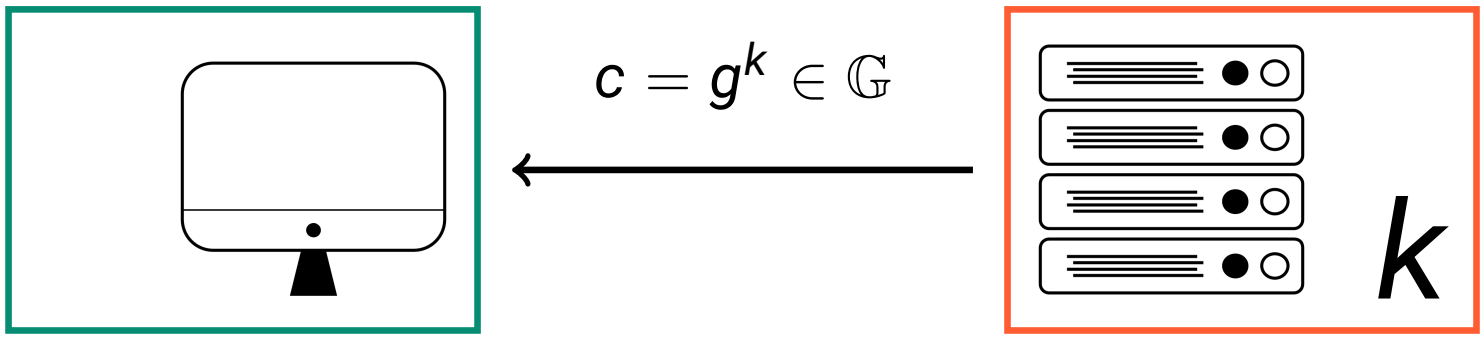
::: Allows mutual client-server authentication secure against pre-computation attacks

::: Compiles an OPRF and authenticated key-exchange to build secure PAKEs

---

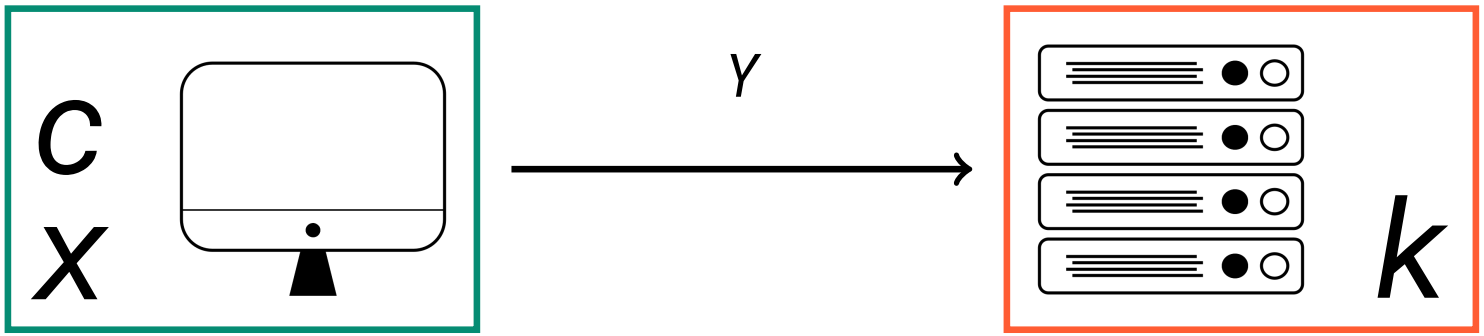
https://datatracker.ietf.org/doc/draft-irtf-cfrg-opaque/

APPLICATIONS: OPAQUE



Stanislaw Jarecki et al. “Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model”. In: IACR Asiacrypt, 2014

CLASSICAL CONSTRUCTION: OFFLINE

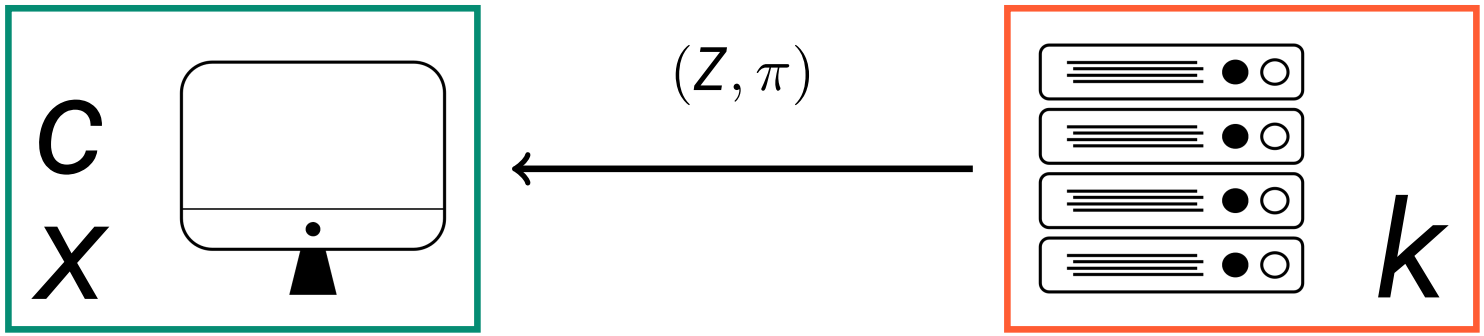


$$X = H(x) \in \mathbb{G}$$

$$Y = X \cdot g^r$$

Stanislaw Jarecki et al. “Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model”. In: IACR Asiacrypt, 2014

CLASSICAL CONSTRUCTION: ONLINE



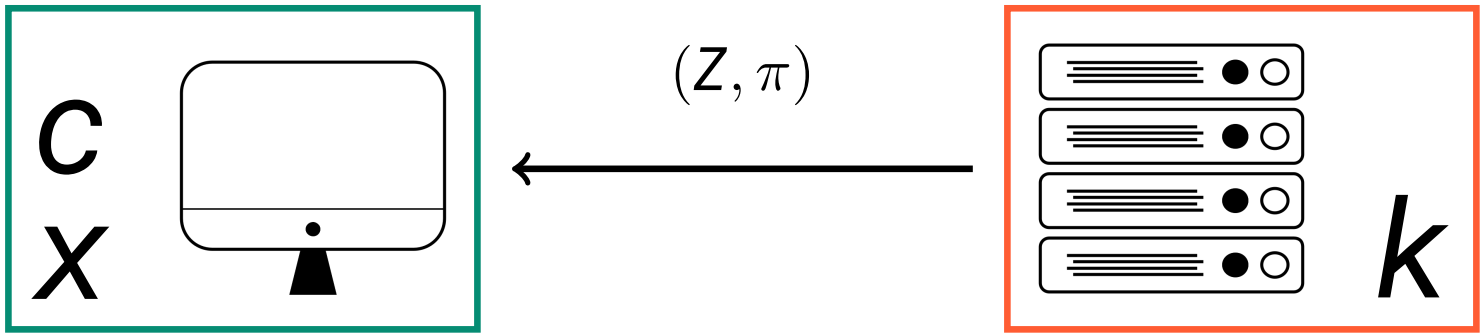
$$Z = Y^k$$

$$\pi = \mathbb{P}(k; c, Y, Z)$$

π proves that c and Z share the same DLOG exponents

Stanislaw Jarecki et al. “Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model”. In: IACR Asiacrypt, 2014

CLASSICAL CONSTRUCTION: ONLINE



$1 =? \mathbb{V}(\pi, c, Y, Z)$

$Z = Y^k$

return $Z \cdot c^{-r}$

$\pi = \mathbb{P}(k; c, Y, Z)$

$Z \cdot c^{-r} = X^k \cdot g^{rk} \cdot (g^k)^{-r} = X^k = \mathsf{PRF}(k, X)$

Stanislaw Jarecki et al. “Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model”. In: IACR Asiacrypt, 2014

CLASSICAL CONSTRUCTION: ONLINE

INTRODUCTION TO VOPRFS AND APPLICATIONS

POST-QUANTUM CONSIDERATIONS

OUR VOPRF CONSTRUCTION

EFFICIENCY AND PARAMETERS

CONCLUSIONS AND OPEN PROBLEMS

::: No previous construction based on lattice-based primitives

::: Boneh et al. (Asiacrypt 20): PQ VOPRFs using isogenies over supersingular elliptic curves

POTENTIAL APPROACHES

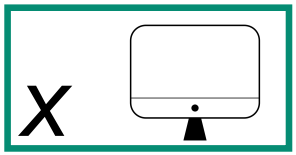
::: (R)LWE-type assumptions provide post-quantum DLOG analogue:

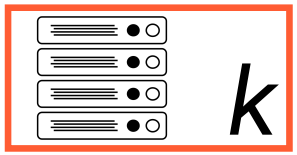
$$\mathsf{k} \in \mathbb{F}_p, \quad (\mathsf{g}, \mathsf{g}^{\mathsf{k}}) \in \mathbb{G} \tag{1}$$

$$\mathsf{s} \in \mathcal{R}, \quad \mathsf{e} \leftarrow\!\!\$\, \mathcal{E}(\mathcal{R}), \quad (\mathsf{a}, \mathsf{a}\,\mathsf{s}+\mathsf{e}) \in \mathcal{R}^2 \tag{2}$$

::: Is it possible to create a natural PQ VOPRF analogue?

STARTING POINTS



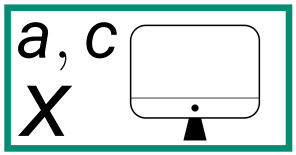


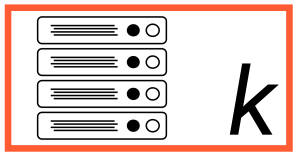
$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE

$$(a, c = a\,k + e)$$

ABSTRACT ATTEMPT: OFFLINE





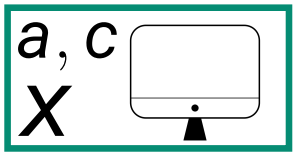
$a_x = \text{encode}(x)$

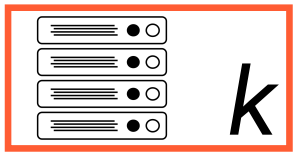
$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE

$$c_x = a_x + a\,s + e_1$$

ABSTRACT ATTEMPT: ONLINE





$a_x = \text{encode}(x)$

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE ZKP

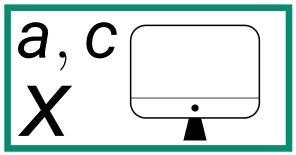
$c_x = a_x + a\,s + e_1$

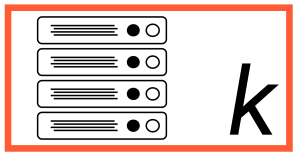
$\longrightarrow$

$(d_x = c_x\,k + e_2, \pi)$

$\longleftarrow$

ABSTRACT ATTEMPT: ONLINE





$a_x = \mathsf{encode}(x)$

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE ZKP

$c_x = a_x + a\,s + e_1$

$\longrightarrow$

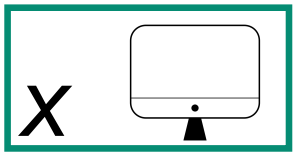
$(d_x = c_x\,k + e_2, \pi)$

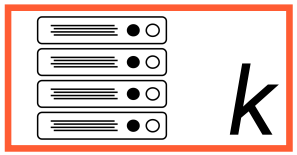
$\longleftarrow$

$$
\begin{aligned}
y_x &= \lfloor d_x - cs \rceil_p \\
&= \lfloor a_x k + ask + e_1 ks + e_2 s - aks \rceil_p \\
&= \lfloor a_x k + e_1 ks + e_2 s \rceil_p \\
&\approx \lfloor a_x k \rceil_p \text{ (when } k, s \text{ are small)}
\end{aligned}
$$

return y_x

ABSTRACT ATTEMPT: ONLINE





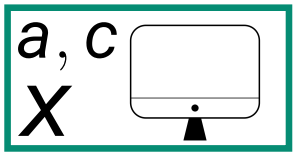
$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

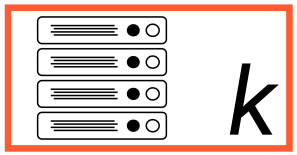
RLWE

How does client verify server commitment?

$(a, c = ak + e)$

PROBLEMS: OFFLINE





$a_x = \mathsf{encode}(x)$ → How do we define encode?

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE ZKP

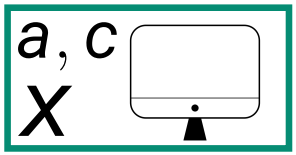
$$\mathsf{c}_x = \mathsf{a}_x + \mathsf{a}\,\mathsf{s} + \mathsf{e}_1 \longrightarrow$$

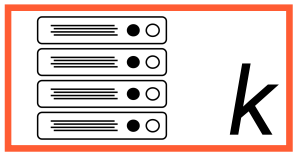
$$\leftarrow (\mathsf{d}_x = \mathsf{c}_x\,\mathsf{k} + \mathsf{e}_2, \pi)$$

$$\begin{aligned} y_x &= \lfloor d_x - cs \rceil_p \\ &= \lfloor a_x k + ask + e_1 ks + e_2 s - aks \rceil_p \\ &= \lfloor a_x k + e_1 ks + e_2 s \rceil_p \\ &\approx \lfloor a_x k \rceil_p \text{ (when } k \text{ is small)} \end{aligned}$$

return y_x

PROBLEMS: ONLINE





$a_x = \text{encode}(x)$ → How do we define encode?

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE ZKP

$c_x = a_x + a\,s + e_1$

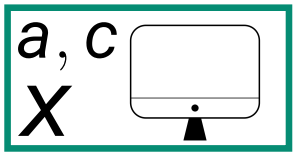
$(d_x = c_x\,k + e_2, \pi)$

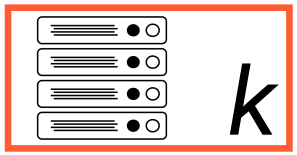
$$
\begin{aligned}
y_x &= \lfloor d_x - cs \rceil_p \\
&= \lfloor a_x k + ask + e_1 ks + e_2 s - aks \rceil_p \\
&= \lfloor a_x k + e_1 ks + e_2 s \rceil_p \\
&\approx \lfloor a_x k \rceil_p \text{ (when } k \text{ is small)}
\end{aligned}
$$

return y_x

How does server verify client input? (input extraction)

PROBLEMS: ONLINE





$a_x = \mathsf{encode}(x)$ → How do we define encode?

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE ZKP

Need to ensure that $(e_2 + ke_1)$ is indistinguishable from a randomly sampled error

$c_x = a_x + a\,s + e_1$ →

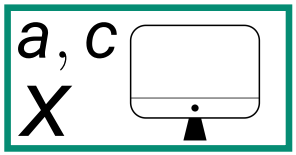
← $(d_x = c_x\,k + e_2, \pi)$

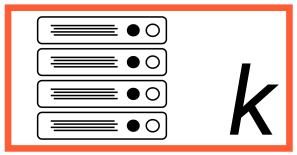
$$
\begin{aligned}
y_x &= \lfloor d_x - cs \rceil_p \\
&= \lfloor a_x k + ask + e_1 ks + e_2 s - aks \rceil_p \\
&= \lfloor a_x k + e_1 ks + e_2 s \rceil_p \\
&\approx \lfloor a_x k \rceil_p \text{ (when } k \text{ is small)}
\end{aligned}
$$

return y_x

How does server verify client input? (input extraction)

PROBLEMS: ONLINE





$a_x = \text{encode}(x)$ → How do we define encode?

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$

RLWE ZKP

Need to ensure that $(e_2 + ke_1)$ is indistinguishable from a randomly sampled error

$c_x = a_x + a\,s + e_1$ →

← $(d_x = c_x\,k + e_2, \pi)$

How to instantiate the required proof system?

$$
\begin{aligned}
y_x &= \lfloor d_x - cs \rceil_p \\
&= \lfloor a_x k + ask + e_1 ks + e_2 s - aks \rceil_p \\
&= \lfloor a_x k + e_1 ks + e_2 s \rceil_p \\
&\approx \lfloor a_x k \rceil_p \text{ (when } k \text{ is small)}
\end{aligned}
$$

return y_x

How does server verify client input? (input extraction)

PROBLEMS: ONLINE

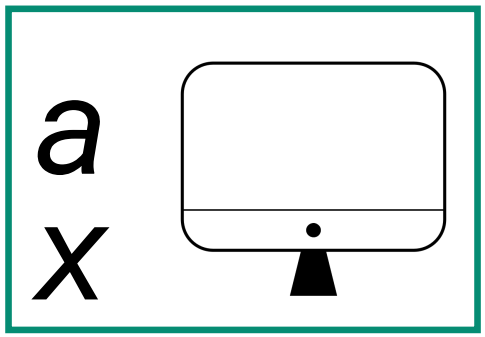
INTRODUCTION TO VOPRFS AND APPLICATIONS

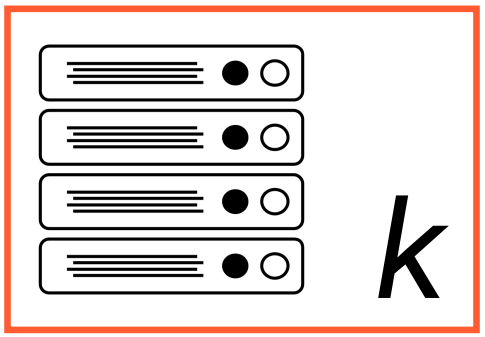
POST-QUANTUM CONSIDERATIONS

OUR VOPRF CONSTRUCTION

EFFICIENCY AND PARAMETERS

CONCLUSIONS AND OPEN PROBLEMS





ZKP_1 PRF

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$ $\mathcal{E}$
RLWE ZKP_0
ZKP_2

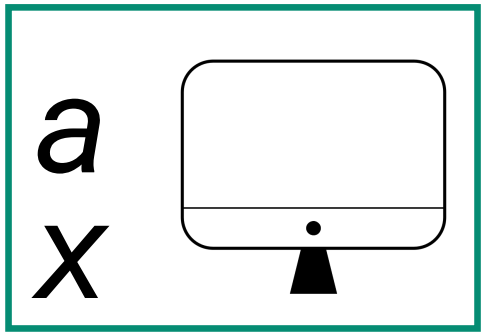
::: $\mathcal{R}$ = $R_q = (\mathbb{Z}_q[\zeta]/\langle\zeta^n + 1\rangle)$, $p|q$ where $p \ll q$

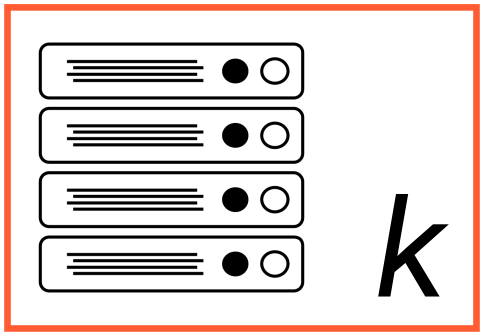
::: $\mathcal{E}$ = $\chi_{\sigma'}$, $\mathcal{K}$ = $\mathcal{E}$ = χ_σ for $\sigma \ll \sigma' \ll q/2p$
- $\mathcal{E}$ must 'drown' samples from $\mathcal{E} \cdot \mathcal{K}$

::: Client and server agree on a $\leftarrow\!\!\$\, \mathcal{R}$

::: PRF is the Banerjee-Peikert (Crypto 14) RLWE-based PRF (with truncated output)

SETUP





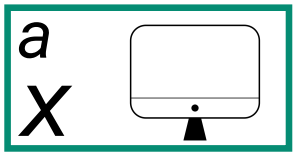
ZKP_1 PRF

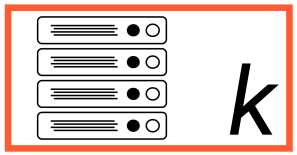
$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$ $\mathcal{E}$
RLWE ZKP_0
ZKP_2

::: ZKP_0: (**Server setup**)
::: ZKP_1: (**Client message**)
::: ZKP_2: (**Server message**)
::: Can use Yang et al. (Crypto 19) or Beullens (Eurocrypt 20) **[QROM compatible]** methods for each ZKP_i

- See paper (Section 4)

PROOF SYSTEMS



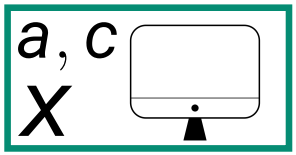


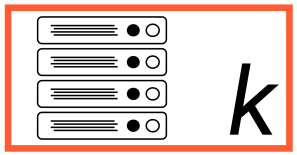
ZKP_1 PRF

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$
$\mathcal{E}$ RLWE
ZKP_0
ZKP_2

$$[c = ak + e, \pi_0]$$

OFFLINE PHASE



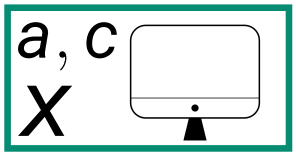


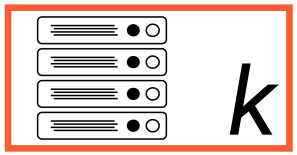
ZKP_1 PRF

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$
$\mathcal{E}$ RLWE
ZKP_0
ZKP_2

$$[c_x = a_x + a\,s + e_1, \pi_1]$$

ONLINE PHASE





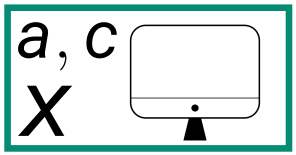
ZKP_1 PRF

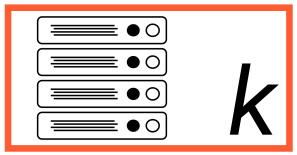
$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$
$\mathcal{E}$ RLWE
ZKP_0
ZKP_2

$$[c_x = a_x + a\,s + e_1, \pi_1] \longrightarrow$$

$$\leftarrow [d_x = c_x\,k + e_2, \pi_2]$$

ONLINE PHASE





ZKP_1 PRF

$\mathcal{R}$ $\mathcal{K}$ $\mathcal{E}$
$\mathcal{E}$ RLWE
ZKP_0
ZKP_2

$$[c_x = a_x + a\,s + e_1, \pi_1] \longrightarrow$$

$$\longleftarrow [d_x = c_x\,k + e_2, \pi_2]$$

$$y_x = \lfloor d_x - cs \rceil_p$$
$$\approx \lfloor a_x k \rceil_p$$

return y_x

ONLINE PHASE

::: Recall protocol output is:

$$y_x = \lfloor a_x k \; + \; e_1 k s \; + \; e_2 s \rceil_p$$

::: We can pick $T = T(\sigma, \sigma', n)$ such that error term is within [-T,T] whp

::: Since k is short, any coefficient of $a_x k$ contained in the set:

$$\{(q/p) \cdot \mathbb{Z} + [-T, T]\},$$

can be used to solve **1D-SIS**

- See paper (Section 3.3) for full details

CORRECTNESS

::: Simulator can extract adversarial secret inputs from accepting ZKPs

::: Send extracted inputs to ideal $\mathcal{F}$

::: Messages are indistinguishable [$\boldsymbol{RLWE}_{q,n,\sigma'}$]

::: Need to prove that correct outputs are generated [**1D-SIS**]

::: Malicious client proof holds in *average-case*

MALICIOUS SECURITY

INTRODUCTION TO VOPRFS AND APPLICATIONS

POST-QUANTUM CONSIDERATIONS

OUR VOPRF CONSTRUCTION

EFFICIENCY AND PARAMETERS

CONCLUSIONS AND OPEN PROBLEMS

	Description	Requirement	Asymptotic
n	ring dimension	$n = \mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$
q	original modulus	$q = p \cdot \sigma' \cdot \lambda^{\omega(1)}$	$\lambda^{\omega(1)}$
p	rounding modulus	–	$\mathsf{poly}(\lambda)$
ℓ	$\log_2(q)$	–	$\omega(1)$
σ	secret/error distribution	$q/\sigma = 2^{o(\sqrt{n})}$	$\mathsf{poly}(\lambda)$
σ'	drowning distribution	$\sigma' = \sigma^2 n^2 \cdot \lambda^{\omega(1)}$	$\lambda^{\omega(1)}$
L	bit-length of PRF input	–	–

ASYMPTOTIC PARAMETERS

For $\lambda \geq 128$:

::: $log_2(q) = 256,\ \sigma = 3.2,\ n = 16,384$
$\Rightarrow$ Ring element $=$ 0.5MB

::: Require $> 2^{40}$ bits of communication per repetition of ZKP proofs[1]
- SNARKs or STARKs would be a better candidate

::: See paper (Section 5.3) for full details

[1] Yang et al. (Crypto 19)

::: **Trapdoors**: Can remove ZKP_0 by trapdooring *a*

::: **Cut-and-choose**: Can remove ZKP_2 at the expense of sending larger client queries

::: **Batching queries**: Can save $O(N \cdot \ell)$ communication over N queries by batching all queries under a single instance of ZKP_2

COMMUNICATION OPTIMISATIONS

Method	Rounds	Concrete	Asymptotic	PQ
JKK14	2	128 bytes	$4\mathbb{G}$	(ROM) $\times$
BKW20	2	$\sim$2MB	$O(\lambda)\mathbb{G}$	(QROM) ✔
Ours	2	$\sim$1MB + **ZKP**	$O(\lambda^{\omega(1)})\mathcal{R} + O(\textbf{ZKP})$	(QROM) ✔

COMPARISON ($\log_2(\lambda) = 128$)

INTRODUCTION TO VOPRFS AND APPLICATIONS

POST-QUANTUM CONSIDERATIONS

OUR VOPRF CONSTRUCTION

EFFICIENCY AND PARAMETERS

CONCLUSIONS AND OPEN PROBLEMS

::: PQ VOPRFs can now be built assuming lattice-based hard problems

::: All PQ proposals suffer from expensive costs due to ZKPs and large parameter settings

::: Future work:

- Can we reduce *all* ZKPs whilst ensuring verifiability?
- Can '*noise drowning*' be avoided?
- Can we use a more efficient PRF alternative?
- Detailed comparison with generic methods?

:: THANKS FOR LISTENING! ::