The return of Eratosthenes: Secure Generation of RSA Moduli using Distributed Sieving

Cyprien Delpech de Saint Guilhem[1], Eleftheria Makri[1], Dragos Rotaru[2], Titouan Tanguy[1]

[1]KU Leuven

[2]Cape Privacy

Summary of Contributions

- RSA modulus generation protocol with generic MPC.
- Up to 37x better communication cost compared to CCD+20.
- Toolbox for MPC over Rings via CRT.
- Convert to Integer protocol, of independent interest.

RSA Modulus

- A biprime N, with two secret prime factors, p and q.

- Heart of the first public key cryptosystem; security based on factoring hardness assumption.

Why RSA Moduli?

- Signatures and Encryption
  - [RSA-77], [Paillier-99].
- Cryptographic accumulators
  - [Benaloh-deMare-93], [Camenisch-Lysyanskaya-02], [Li-Li-Xue-07], [Boneh-Bünz-Fisch-19],
- VDF and Timelock puzzles
  - [Rivest-Shamir-Wagner-99], Boneh-Bonneau-Bünz-Fisch-18], [Wesolowski-19], [Pietrzak-19], [Ephraim-Freitag-Komargodski-Pass-19].
- Efficient zk-SNARKs
  - [Bünz-Fisch-Szepieniec-19], [Lai-Malavolta-19]
- And others...

Why (distributed) RSA Moduli?

- Threshold Cryptography

Call 2021a for Feedback on Criteria for Threshold Schemes

NIST Multi-party Threshold Cryptography

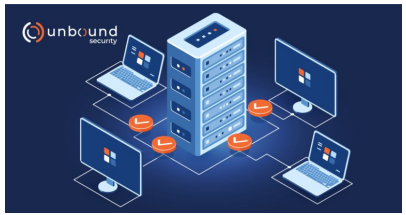
2021-July-02: https://csrc.nist.gov/projects/threshold-cryptography

Please send comments to threshold-MP-call-2021a@nist.gov by September 13, 2021.

1. Scope of proposals. The future call for proposals will be intended to gather expert submissions of concrete threshold schemes for primitives that are *interchangeable* (in the sense of IR 8214A, Section 2.4) with[2] ECDSA, EdDSA, RSA signing/decryption, RSA keygen, AES, and ECC-based key agreement.[3] After an evaluation period, and possibly various stages for tweaks,

Why (distributed) RSA Moduli?

- *Companies or foundations*



VDF Alliance

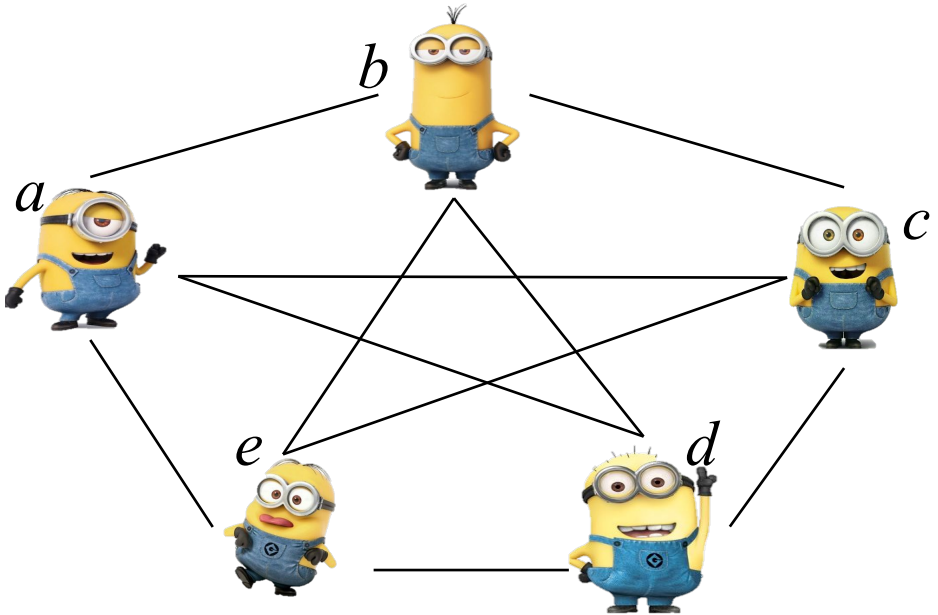
Our main result

Up to 37x better communication cost compared to CCD+20.

- our RSA modulus generation works with ANY LSSS based MPC.
- along the way we had to develop a toolbox for MPC operations over CRT…

Main tool

- Generic multiparty computation
- Work with CRT components



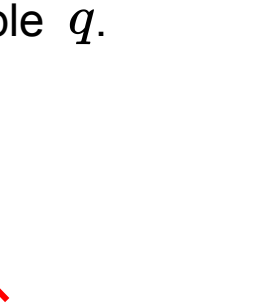
Securely compute $f(a, b, c, d, e)$.

Textbook RSA modulus generation

1. Choose random $p \leftarrow \mathbb{Z}_{2^k}$
2. If p is not prime return to Step 1.
3. Repeat first two steps to sample q.
4. Compute $N = p \cdot q$.

Textbook RSA modulus generation

1. Choose random $p \leftarrow \mathbb{Z}_{2^k}$
2. If p is not prime return to Step 1.
3. Repeat first two steps to sample q.
4. Compute $N = p \cdot q$.



$a^{p-1} \bmod p$

Distributed RSA modulus generation

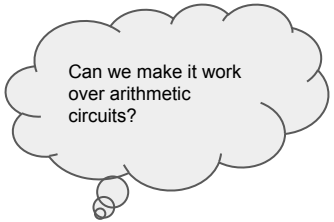
1. Sample p, q as integer shares.
2. Compute $N = p \cdot q$
3. Check whether N is bi-prime using p, q .
4. Parties run a consistency check to protect from malicious behaviour.

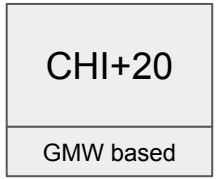
Related Work

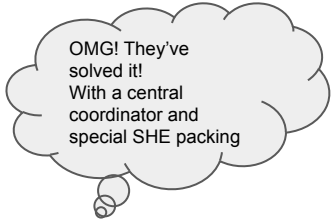
Protocol	Security	Dishonest Majority	#Parties	Test	Leakage-free
[BF97]	Passive	✗	$n \geq 3$	biprimality	✓
[FMY98]	Active	✗	$n \geq 3$	biprimality	✓
[PS98]	Active	✓	$n = 2$	biprimality	✗
[Gil99]	Passive	✓	$n = 2$	biprimality	✓
[ACS02]	Passive	✗	$n \geq 3$	primality	✓
[DM10]	Active	✗	$n = 3$	primality	✓
[HMRT12, HMR+19]	Active	✓	$n \geq 2$	biprimality	✓
[FLOP18]	Active	✓	$n = 2$	biprimality	✗
[CCD+20]	Active	✓	$n \geq 2$	biprimality	✓
[CHI+20]	Active*	✓	$n \geq 2$	biprimality	✓
Ours	Active	✓	$n \geq 2$	biprimality	✓

Connections with related work

Connections with related work







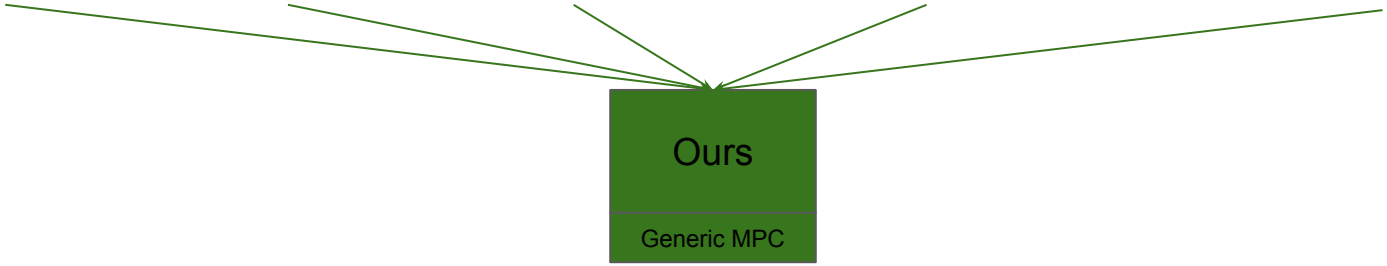
BonehFranklin97
Protocol Blueprint

MalkinWuBoneh99
Distributed sieving

DamgårdMikkelsen2010
Integer sharing

GRS+16 (CCS)
Malicious exponentiation

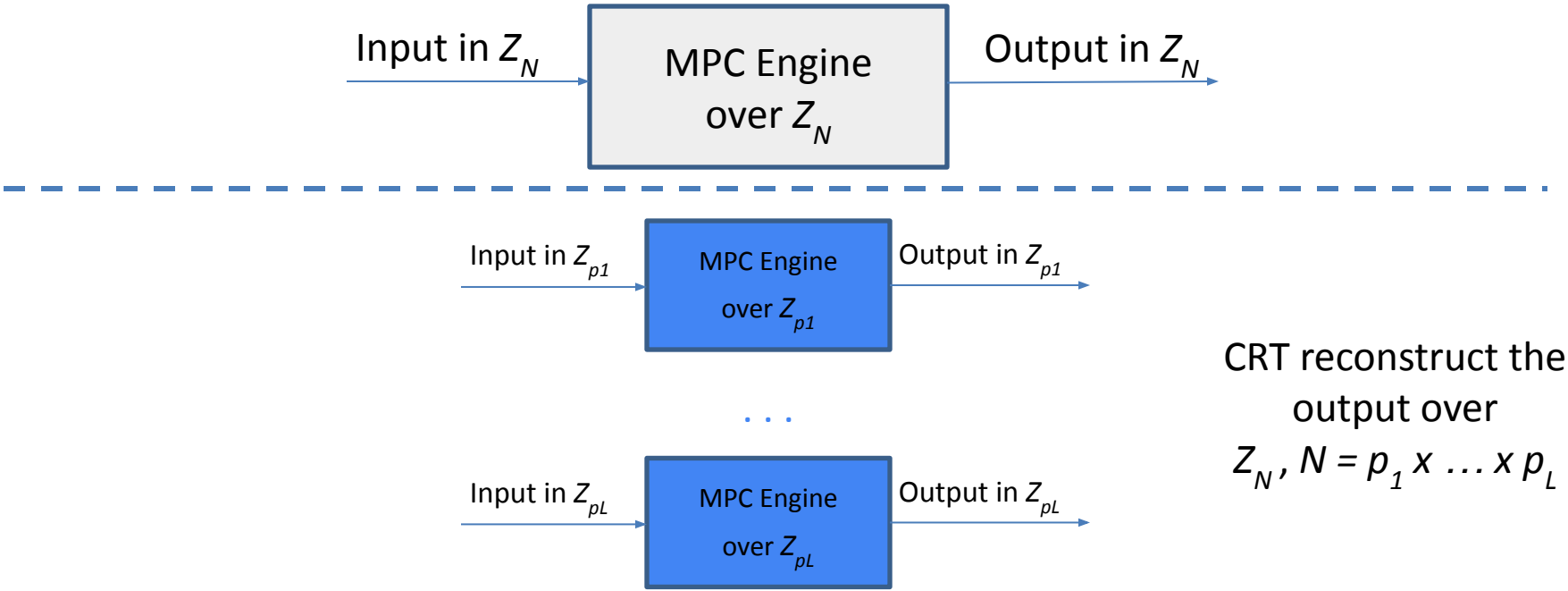
RST+19
Core of malicious check



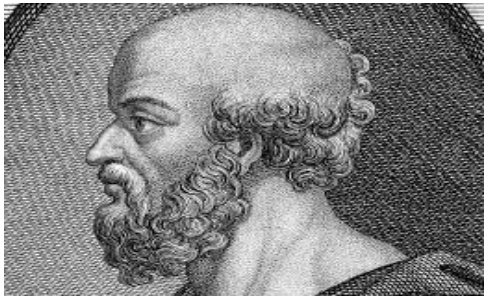
Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\,q$ and reveal N
3. Jacobi biprimality test
4. Consistency check
5. GCD test

MPC – CRT



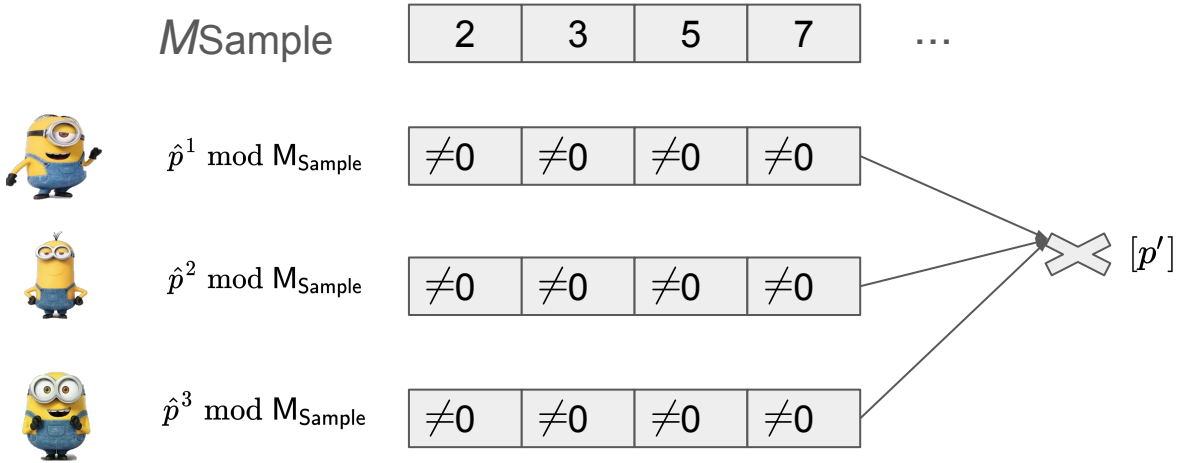
CRT reconstruct the output over Z_N, $N = p_1 \, x \, \ldots \, x \, p_L$

The  of 

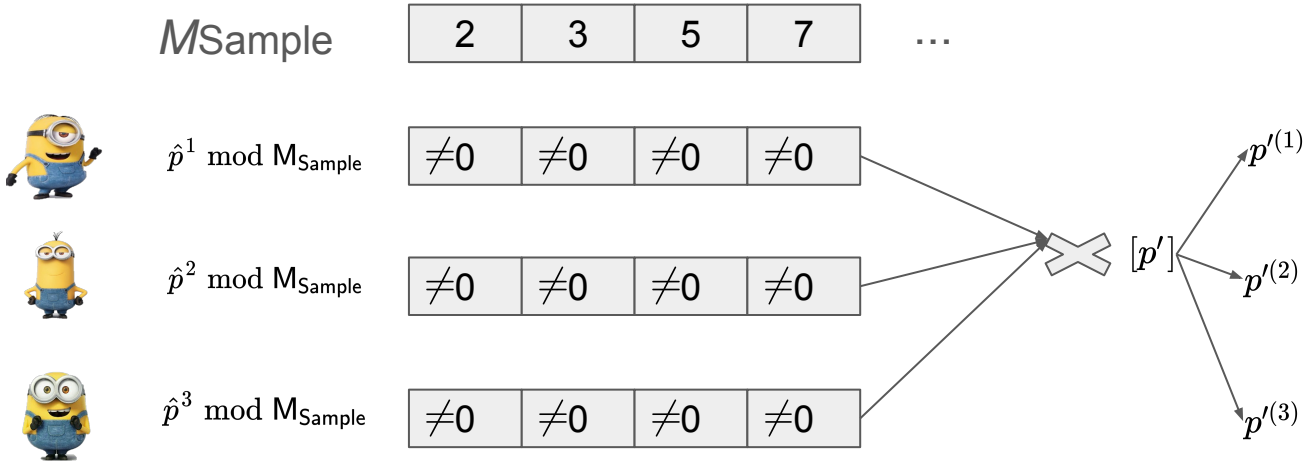
Distributed Sieving

*M*Sample	2	3	5	7	…
 $\hat{p}^1 \bmod M_{Sample}$	≠0	≠0	≠0	≠0	
 $\hat{p}^2 \bmod M_{Sample}$	≠0	≠0	≠0	≠0	
 $\hat{p}^3 \bmod M_{Sample}$	≠0	≠0	≠0	≠0	

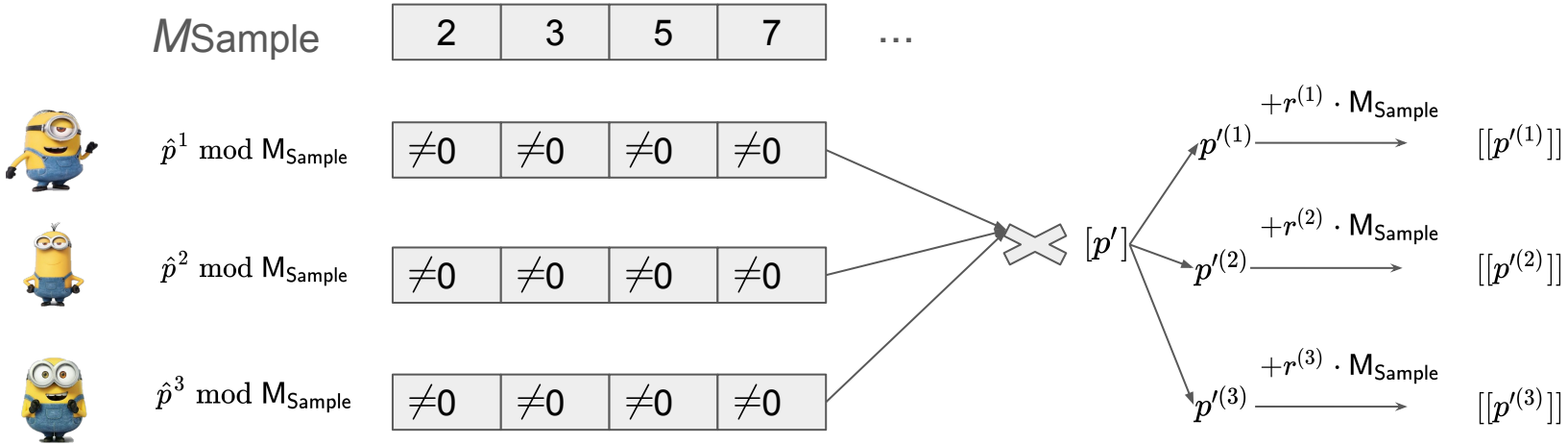
Distributed Sieving



Distributed Sieving



Distributed Sieving



Our Protocol

1. Sample candidate primes p and q
2. **Securely compute $N = p\,q$ and reveal N**
3. Jacobi biprimality test
4. Consistency check
5. GCD test

Combine

$[[p]]$ $[[q]]$

$[[p'^{(1)}]]$ $[[q'^{(1)}]]$

$+$ $+$

$[[p'^{(2)}]]$ $[[q'^{(2)}]]$

$+$ $+$

$[[p'^{(3)}]]$ $[[q'^{(3)}]]$

Prevent overflow

$$N = Open([[p]] \cdot [[q]])$$

Combine

- Extend the CRT representation: product is taken over the integers (i.e., prevent overflow).
- Perform “standard” secure multiplication over the MPC-CRT engines
- Reveal and CRT-Reconstruct the product N
- Check that N falls within the predetermined bounds, and is coprime to M_{Sample}

Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\,q$ and reveal N
3. **Jacobi biprimality test**
4. Consistency check
5. GCD test

Jacobi Test

- Sample public $\gamma \in \mathbb{Z}_N$ s.t. the Jacobi symbol $(\frac{\gamma}{N}) = 1$
- Securely compute $\phi(N)/4$ in the exponent of γ
- Abort if $\gamma^{\phi(N)/4} \neq \pm 1$

- *This test accepts false positives with probability ½. We repeat the test* sec *times to increase the probability of N being a biprime to* 2^{-sec}.

Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\,q$ and reveal N
3. Jacobi biprimality test
4. **Consistency check**
5. GCD test

Consistency Check

- *This check ensures security against malicious parties, who contributed inconsistent shares to the Jacobi test.*

1. LevelUp s.t. the CRT representation allows the consistency check computations to be performed without overflow.
2. Sample bounded randomness and multiplicatively mask the secret exponent
3. Convert the CRT represented masked sharing to a sharing over the integers

From CRT share to Integer share

$[[x_{\mathsf{CRT}}]]$

$x_{\mathsf{CRT}}^{(1)}$

$x_{\mathsf{CRT}}^{(2)}$

$x_{\mathsf{CRT}}^{(3)}$

From CRT share to Integer share

	$[[x_{\mathsf{CRT}}]]$	$[[r_{\mathsf{CRT}}]], [r_{\mathsf{Int}}]$	
	$x^{(1)}_{\mathsf{CRT}}$	$r^{(1)}_{\mathsf{CRT}}, r^{(1)}_{\mathsf{Int}}$	
	$x^{(2)}_{\mathsf{CRT}}$	$r^{(2)}_{\mathsf{CRT}}, r^{(2)}_{\mathsf{Int}}$	$t \leftarrow Open([[x_{\mathsf{CRT}}]] + [[r_{\mathsf{CRT}}]])$
	$x^{(3)}_{\mathsf{CRT}}$	$r^{(3)}_{\mathsf{CRT}}, r^{(3)}_{\mathsf{Int}}$	

From CRT share to Integer share

	$[[x_{\mathsf{CRT}}]]$	$[[r_{\mathsf{CRT}}]], [r_{\mathsf{Int}}]$		$[x_{\mathsf{Int}}]$
	$x^{(1)}_{\mathsf{CRT}}$	$r^{(1)}_{\mathsf{CRT}}, r^{(1)}_{\mathsf{Int}}$		$x^{(1)}_{\mathsf{Int}} = t - r^{(1)}_{\mathsf{Int}}$
	$x^{(2)}_{\mathsf{CRT}}$	$r^{(2)}_{\mathsf{CRT}}, r^{(2)}_{\mathsf{Int}}$	$t \leftarrow Open([[x_{\mathsf{CRT}}]] + [[r_{\mathsf{CRT}}]])$	$x^{(2)}_{\mathsf{Int}} = t - r^{(2)}_{\mathsf{Int}}$
	$x^{(3)}_{\mathsf{CRT}}$	$r^{(3)}_{\mathsf{CRT}}, r^{(3)}_{\mathsf{Int}}$		$x^{(3)}_{\mathsf{Int}} = t - r^{(3)}_{\mathsf{Int}}$

Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\,q$ and reveal N
3. Jacobi biprimality test
4. Consistency check
5. **GCD test**

Efficiency Analysis (1/2)

Scheme	CCD+20	Ours	CCD+20	Ours	CCD+20	Ours
κ	**1024**	**1024**	**1536**	**1536**	**2048**	**2048**
semi-honest (MB)	139	41.68	416	116.55	910	243.3
malicious (GB)	20.81	0.64	43.42	1.188	74.52	1.99

Communication cost per party, for 2-party protocol.

Efficiency Analysis (2/2)

Scheme	CCD+20	Ours	CCD+20	Ours	CCD+20	Ours
κ	**1024**	**1024**	**1536**	**1536**	**2048**	**2048**
semi-honest (MB)	2.09	4.34	6.24	12.17	13.65	25.23
malicious (GB)	1020	68.8	4734	153.2	8100	281.91

Communication cost per party, for 16-party protocol.

Summary of Contributions

- RSA modulus generation protocol with generic MPC.
- Exploit *Distributed Sieving techniques* and *public knowledge* to perform it semi-honestly without degrading security.
- Convert to Integer protocol, of independent interest.
- Up to 37x better communication cost compared to CCD+20.

Thank you!