DualRing: Generic Construction of Ring Signatures with Efficient Instantiations

Tsz Hon Yuen
The University of Hong Kong, Hong Kong

Muhammed F. Esgin
Monash University, CSIRO's Data61 Australia

Joseph K. Liu
Monash University, Australia

Man Ho Au
The University of Hong Kong, Hong Kong

Zhimin Ding
Rice University, USA

CRYPTO 2021
2021/8/17

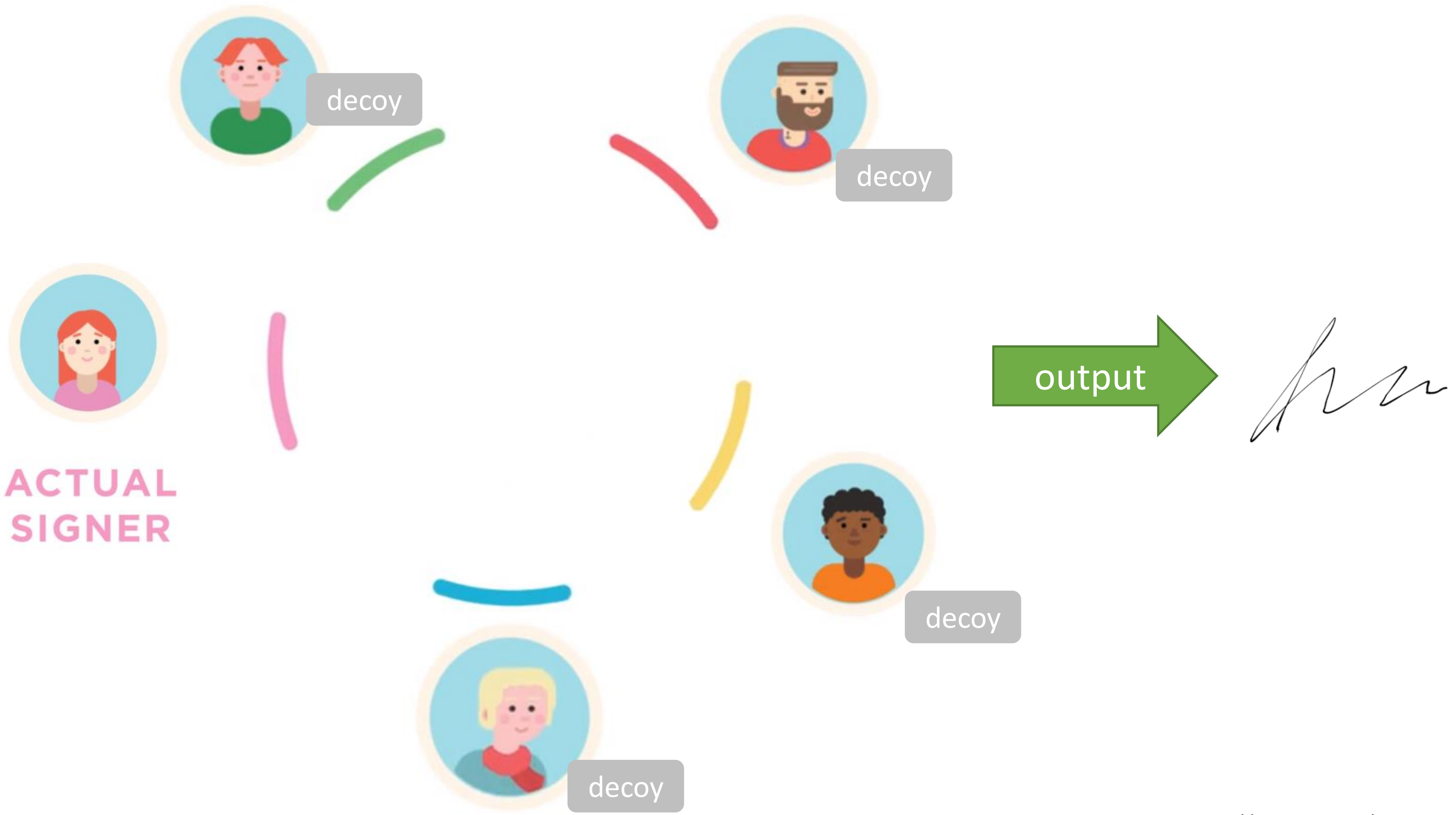
Table of Contents

Introduction

Ring Signature

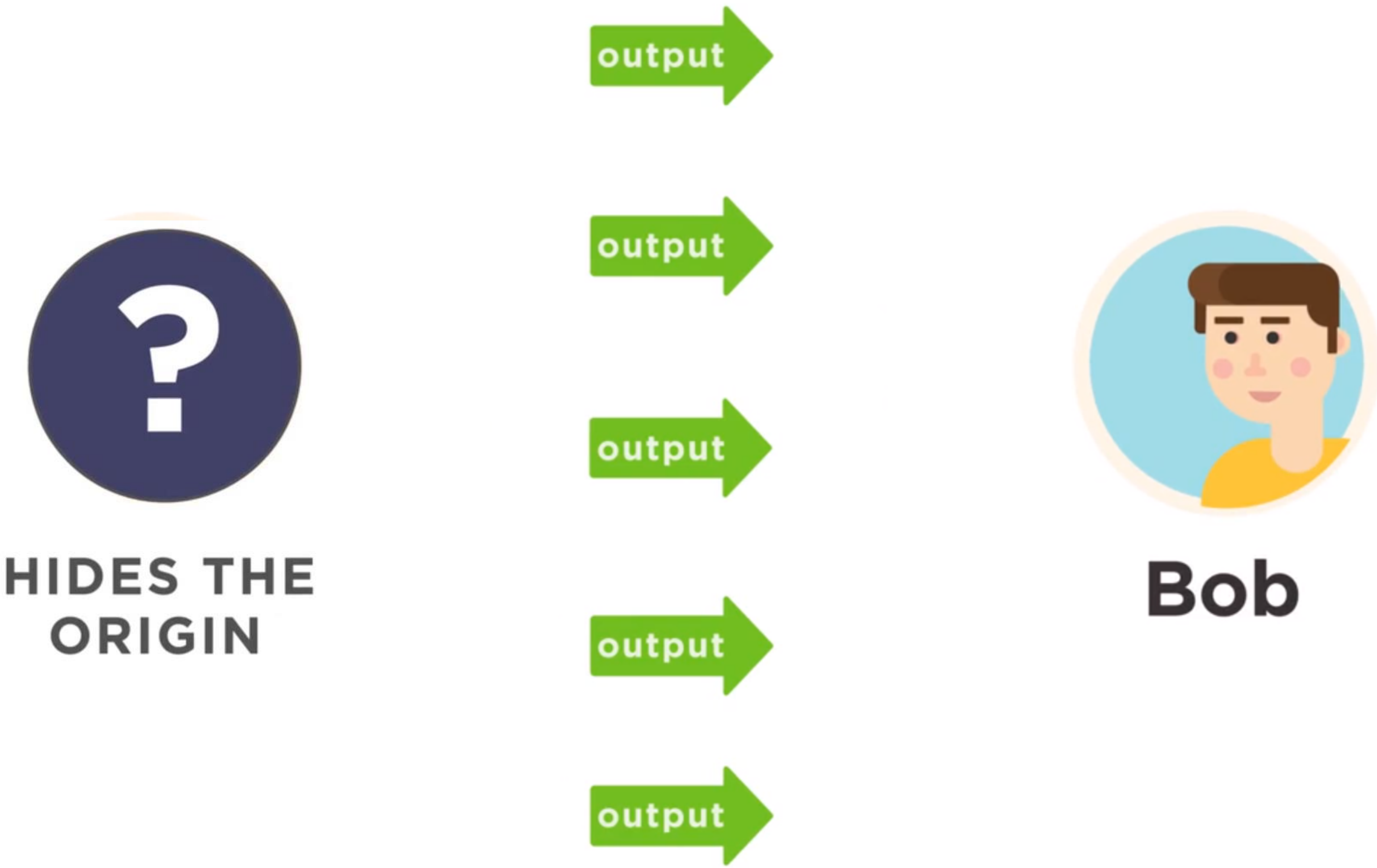
Introduction

- Ring Signature



Source: Monero Brasil. https://vimeo.com/233677706

Introduction



Source: Monero Brasil. https://vimeo.com/233677706

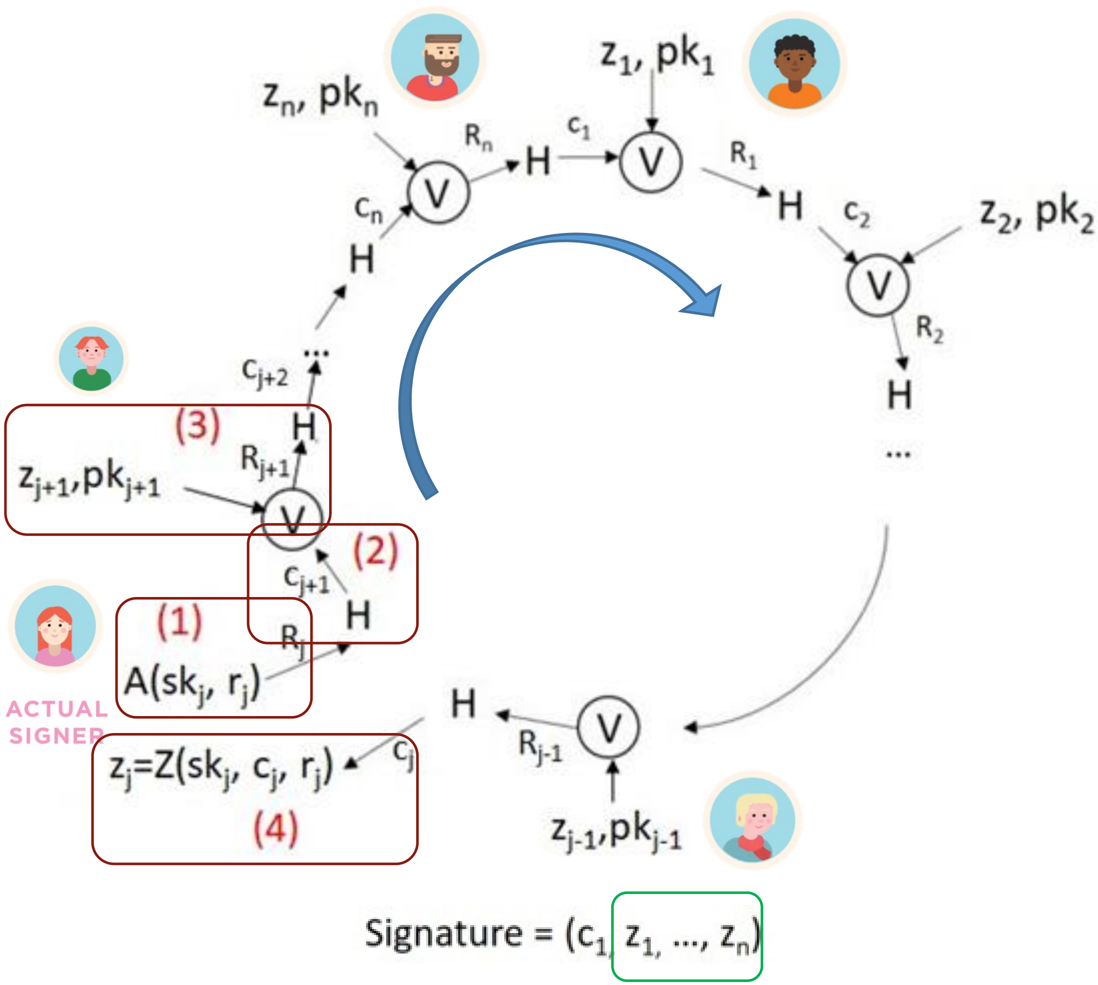
Classical Ring Signature

- [RST01], [AOS02]
- Recall: Type-T (Three-move) Signature (e.g. Schnorr)
  1. a commit function A, which outputs a commitment R
     - $A(r) \rightarrow R := g^r$
  2. a hash function H, which outputs a challenge c
     - $H(M, R) \rightarrow c$
  3. a response function Z, which outputs a response z
     - $Z(r, c, sk) \rightarrow z = r + c \cdot sk$
  4. a verification function V, which reconstruct R from (c,z) and then runs H to check if c is correct
     - $V(pk, c, z) \rightarrow R = g^z \cdot pk^{-c},\ c = H(M, R)$

[RST01] Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT 2001.
[AOS02] Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: ASIACRYPT 2002.

Classical Ring Signature

- [AOS02]



1) Signer picks r_j to generate R_j via the commit function A
2) Signer uses R_j to compute the (j+1)-th challenge c_{j+1} by the hash function H
3) Pick a random response z_{j+1} and the *j+1*-th user pk_{j+1}, signer reconstruct the R_{j+1} using the verify function V and then generate c_{j+2} by H.
   - A ring is then formed sequentially.
4) The last step is to compute z_j from sk_j, c_j, r_j using the response function Z

Current Ring Signature Schemes

Accumulator-Based

- √ Constant signature size
- X Trusted setup: RSA/pairing-based
- X Large signature size: lattice-based (~ several MBs)

ZK Proof-Based

- √ O(log n) size
- One-out-of-many proof/Bulletproof: DL-based [GK15], lattice-based [ESLL19]

[GK15] Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: EUROCRYPT 2015.
[ESLL19] Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: CRYPTO 2019.

Can we do better?

- DL-based:

Ring Signatures	# elements in signature		Signature Size (Bytes)				
	$\mathbb{G}$	$\mathbb{Z}_p$	$n=2$	$n=8$	$n=64$	$n=2048$	$n=4096$
LPQ@ESORICS18	$4\log n + 2$	$5\log n + 4$	480	1070	1946	3114	3406
GK@Eurocrypt15	$4\log n$	$3\log n + 1$	260	716	1400	2540	2768
BCC+@ESORICS15	$\log n + 12$	$\frac{3}{2}\log n + 6$	669	831	1074	1479	1560
YSLAEZ@FC20	$2\log n + 7$	7	521	653	851	1181	1247
LRR+@CCS19	$2\log(n+2) + 4$	5	424	523	721	1051	1117
DualRing-EC	$2\log n + 1$	3	195	327	525	855	921

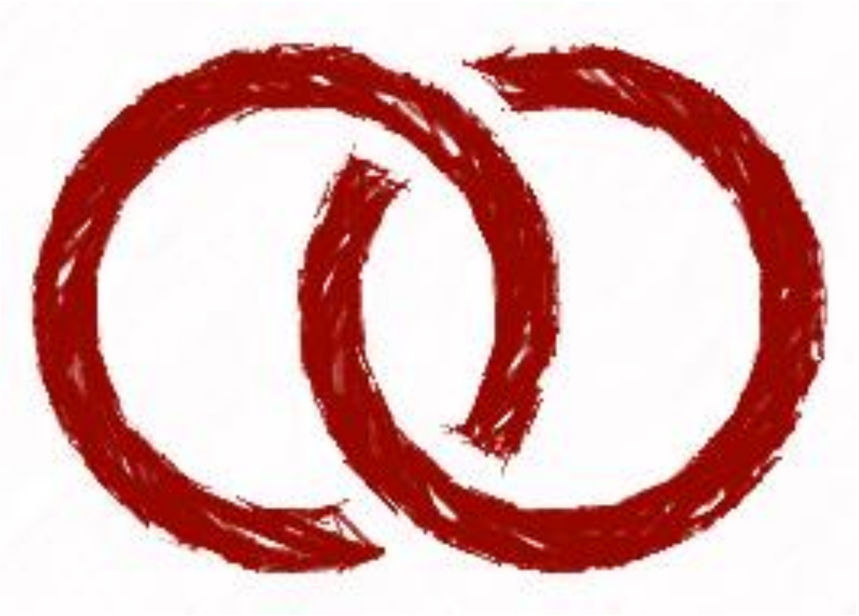
- Lattice-based:

Ring Signatures	Signature Size (Bytes)						Assumption
	$n=2$	$n=8$	$n=64$	$n=1024$	$n=2048$	$n=4096$	
LAZ@ACNS19	2532	10128	81024	1296384	2592768	6564888576	NTRU
BKP@Asiacrypt20	49000	50000	52000	54000	54500	55000	M-LWE+M-SIS
EZSLL@CCS19	18000	19000	31000	48000	53000	59000	M-LWE+M-SIS
DualRing-LB	4480	4630	6020	31160	55500	106570	M-LWE+M-SIS

DualRing: Generic construction from the classical ring approach!

DualRing

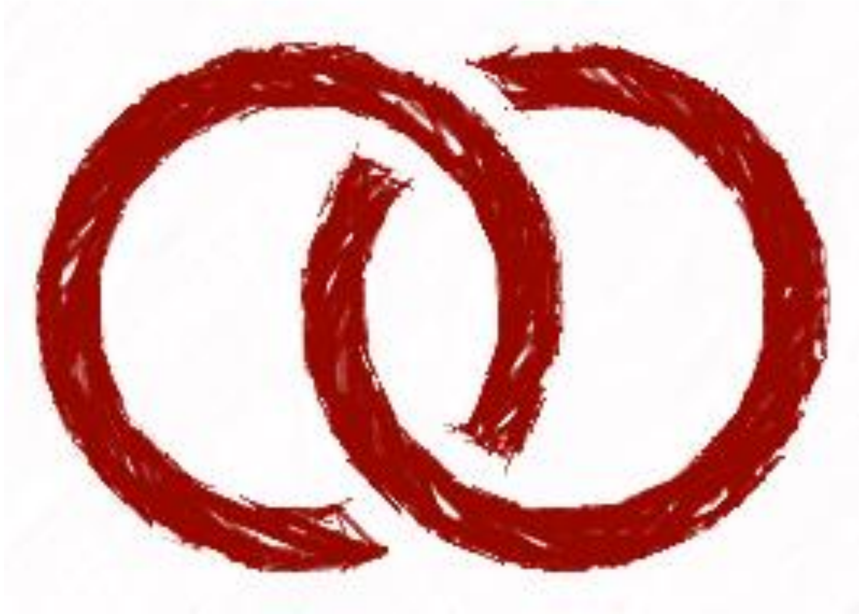
High level idea



Overview of DualRing

- Revisit the classical ring approach and make it shorter (i.e., O(log n) size)
  - [AOS02] includes the hash function H in the ring structure
    → difficult to shorten the signature
- Goal: formation of rings with simple algebraic operation
  - Ring structure provides anonymity as [RST01], [AOS02]
  - Simple algebraic operations (i.e., without H) allow compression

Ring of
commitments



Ring of
challenges

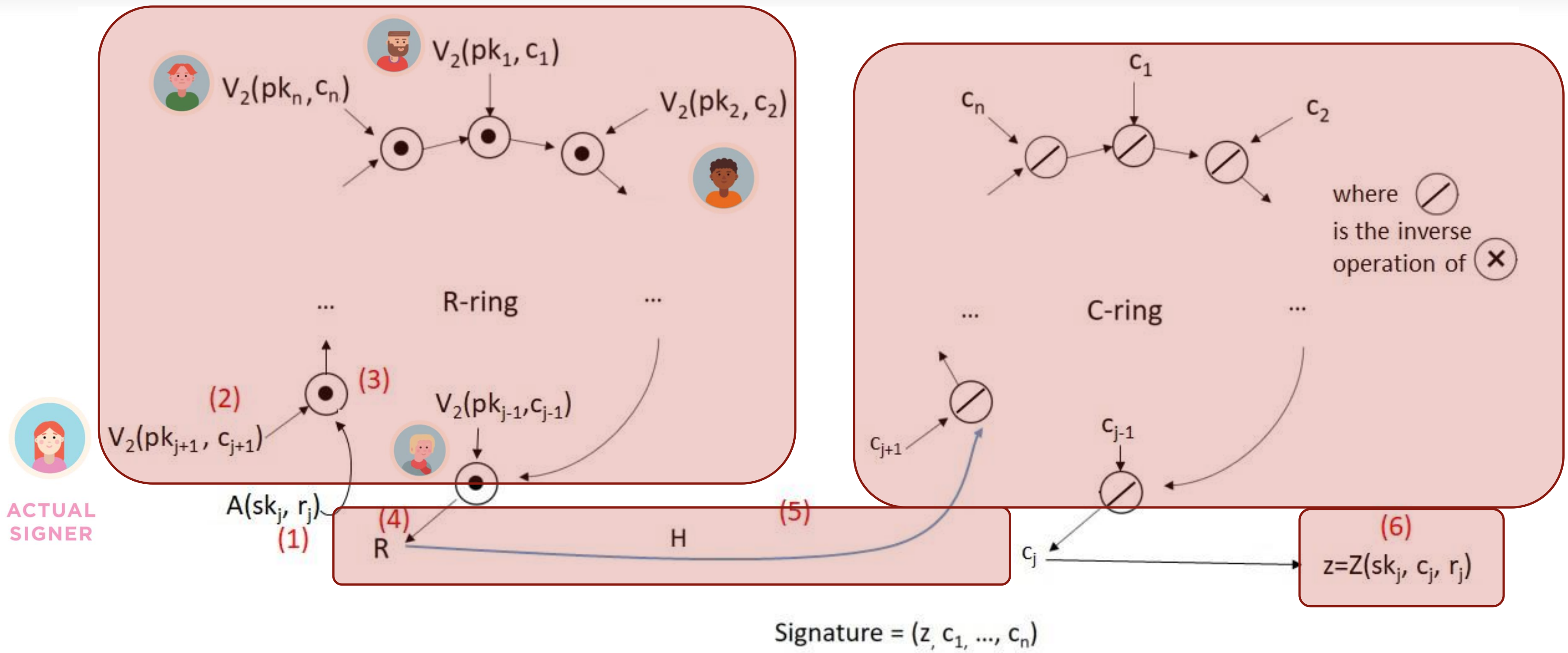
Building Block

- Type-T* canonical identification (Three-move)
  - The verify function V can be written as:
    $V(pk, c, z) = V_1(z) \odot V_2(pk, c)$ $\qquad V(pk, c, z) \rightarrow R = g^z \cdot pk^c$
  - V_1 is additive/multiplicative homomorphic
  - Given sk and c, there exists a function $T(sk, c) \rightarrow z'$ such that $\mathrm{V}_1(\mathrm{z}') = \mathrm{V}_2(pk, c)$
  - The challenge space is a group
  - Has special soundness*

> - Schnorr identification, identification scheme from Katz-Wong signature, Chaum-Pedersen identification and Okamoto-Schnorr identification
> - GQ identification
> - Lattice-based identification from "Fiat-Shamir with Aborts"

* In the paper, we use a new notion called *special impersonation* to deal with the "knowledge gap" in the lattice-based setting

Construction

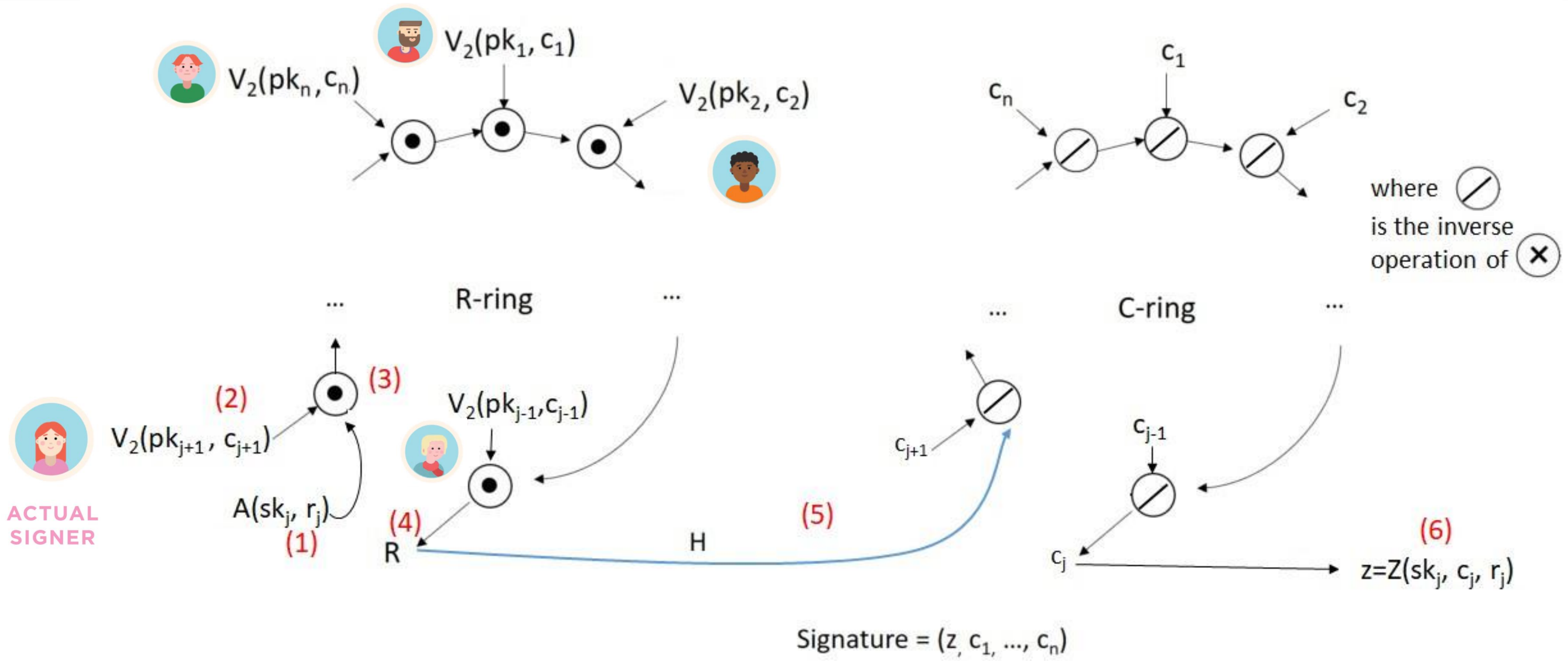


1) Signer picks r_j
2) Signer picks random challenges $c_1, \dots, c_{j-1}, c_{j+1}, \dots, c_n$
3) Form an R-ring by:

R= $A(r_j) \odot V_2(pk_{j+1}, c_{j+1}) \dots \odot V_2(pk_n, c_n) \odot V_2(pk_1, c_1) \dots \odot V_2(pk_{j-1}, c_{j-1})$

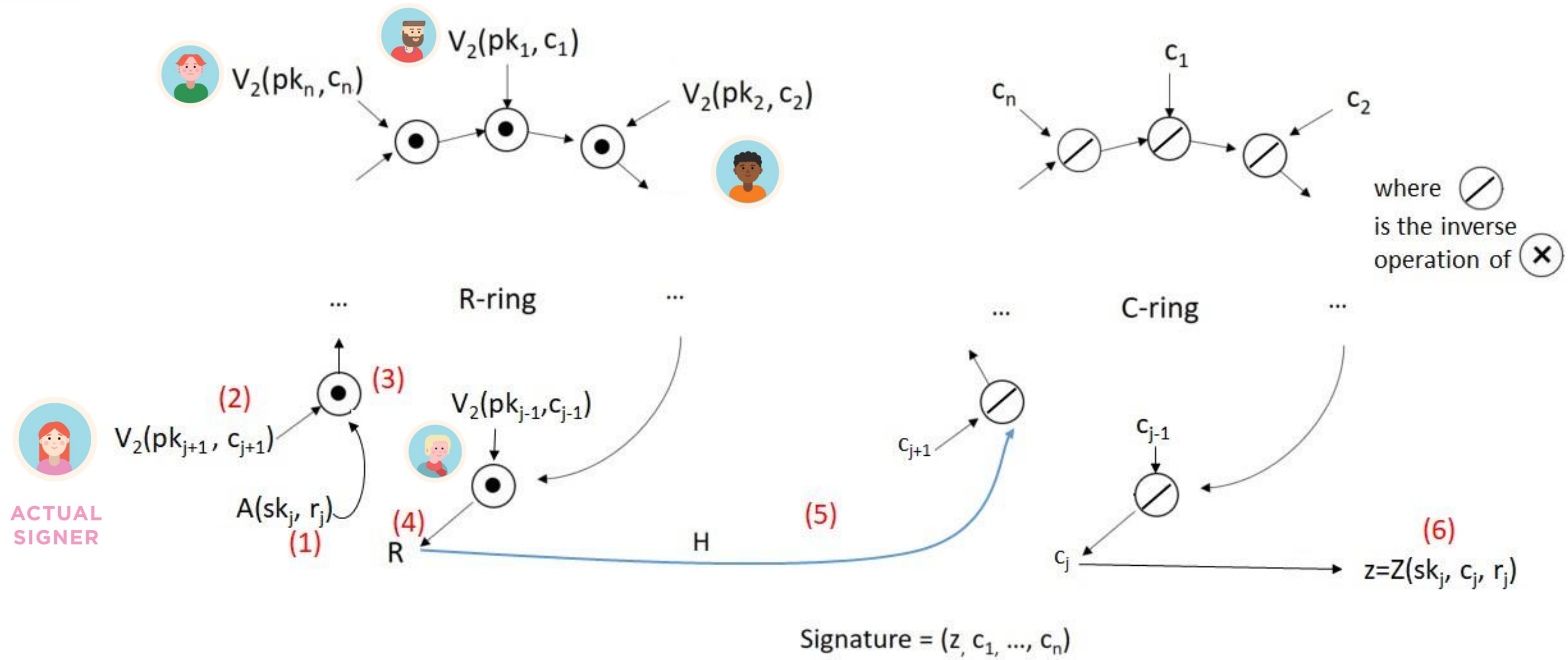
4) Use R to compute c = H(R, {pk}, m)
5) Form C-ring, by computing c_j where:
$$c = c_1 \otimes \cdots \otimes c_n$$
6) Compute $z = Z(sk_j, c_j, r_j)$

Construction



- Verification:
  - $R = V_1(z) \odot \bigodot_{i=1}^{n} V_2(pk_i, c_i)$
  - Check if $c = H(R, \{pk\}, M) = c_1 \otimes \cdots \otimes c_n$

Security

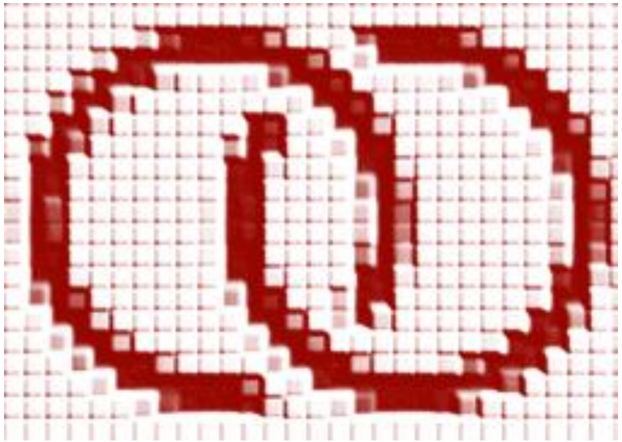


Anonymity:
- Provided by the ring structure in the ROM as [RST01], [AOS02]

Unforgeability:
- Reduced to the security of Type-T* canonical identification in the ROM

Instantiations

ECC-based and lattice-based



DualRing-EC

- DualRing only gives a O(n)-size generic ring signature $(z, c_1, \ldots, c_n)$
- In the DL-based setting, we have

  $c = c_1 + c_2 + \ldots + c_n \quad \text{mod p}$

- We propose a (non-interactive) *sum argument of knowledge (NISA)* to compress the challenges
  - Similar to the Bulletproof (for inner product relation), our NISA saves about half of the computation due to the simplicity of the sum relation.

Algorithm 4: NISA

Procedure NISA.$\textsc{Proof}(\{\mathsf{param}, \mathbf{g}, P, c\}, \mathbf{a})$:
  Run protocol $\textsc{Pf}$ on input $(\mathbf{g}, u^{H'_Z(P,u,c)}, \mathbf{a}, \mathbf{1}^n)$;

Procedure $\textsc{Pf}(\mathbf{g}, \hat{u}, \mathbf{a}, \mathbf{b})$:
  // Assume **L**, **R** are initially empty, but maintains its memory throughout the recurrsion. n is the length of vector **a** and **b**.
  if $n = 1$ **then**
    Output $\pi = (\mathbf{L}, \mathbf{R}, a, b)$.
  else
    Compute $n' = \frac{n}{2}$, $c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p$, $c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p$;
    $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \hat{u}^{c_L} \in \mathbb{G}$ and $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \hat{u}^{c_R} \in \mathbb{G}$;
    Add L to **L** and R to **R** and compute $x = H_Z(L, R)$;
    Compute $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$, $\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$ and $\mathbf{b}' = x^{-1} \cdot \mathbf{b}_{[:n']} + x \cdot \mathbf{b}_{[n':]} \in \mathbb{Z}_p^{n'}$;
    Run protocol $\textsc{Pf}$ on input $(\mathbf{g}', \hat{u}, \mathbf{a}', \mathbf{b}')$;

Procedure NISA.$\textsc{Verify}(\mathsf{param}, \mathbf{g}, P, c, \pi = (\mathbf{L}, \mathbf{R}, a, b))$:
  $P' = P \cdot u^{c \cdot H'_Z(P,u,c)}$;
  Compute for all $j = 1, \ldots, \log_2 n$: $x_j = H_Z(L_j, R_j)$;
  Compute for all $i = 1, \ldots, n$:
  $y_i = \prod_{j \in [\log_2 n]} x_j^{f(i,j)}, f(i,j) = \begin{cases} 1 & \text{if } (i-1)\text{'s } j\text{-th bit is } 1 \\ -1 & \text{otherwise} \end{cases};$
  Set $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{x} = (x_1, \ldots, x_{\log_2 n})$;
  if $\mathbf{L}^{\mathbf{x}^2} P' \mathbf{R}^{\mathbf{x}^{-2}} = \mathbf{g}^{a \cdot \mathbf{y}} u^{ab \cdot H'_Z(P,u,c)}$ **then**
    Output 1
  Output 0

DualRing-EC

- Combining DualRing with NISA, we get the shortest ring signature

Algorithm 5: DualRing-EC

Procedure $\textsc{Setup}(\lambda)$:
- $\mathsf{param}' \leftarrow \textsc{DualRing.Setup}(\lambda)$;
- pick a generator $u \leftarrow_s \mathbb{G}$;
- return $\mathsf{param} = (\mathsf{param}', u)$;

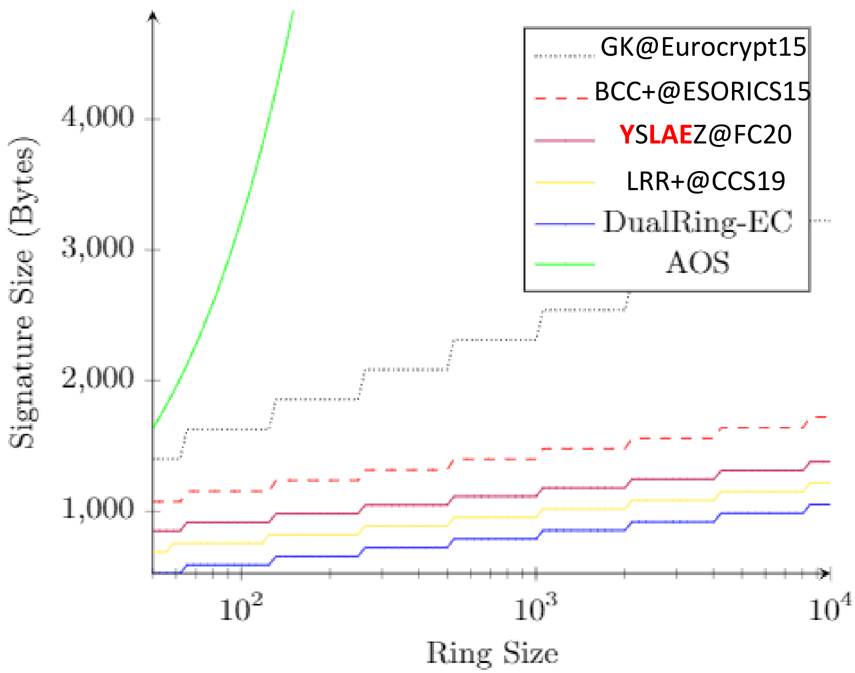
Procedure $\textsc{Sign}(\mathsf{param}, m, \mathbf{pk}, sk_j)$:
- $(c_1, \ldots, c_n, z) \leftarrow \textsc{DualRing.Sign}(\mathsf{param}, m, \mathbf{pk}, sk_j)$;
- `// (c, R) is computed in` $\textsc{DualRing.Sign}$
- $\mathbf{a} \leftarrow (c_1, \ldots, c_n)$;
- $P = R \cdot (V_1(z))^{-1}$;
- $\pi \leftarrow \text{NISA.}\textsc{Proof}(\{\mathsf{param}, \mathbf{pk}, u, P, c\}, \mathbf{a})$;
- return $\sigma = (z, P, \pi)$;

Procedure $\textsc{KeyGen}(\mathsf{param})$:
- return $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{DualRing.KeyGen}(\mathsf{param}')$;

Procedure $\textsc{Verify}(\mathsf{param}, m, \mathbf{pk}, \sigma)$:
- parse $\sigma = (z, P, \pi)$;
- $R = P \cdot V_1(z)$;
- $c = H_Z(m, \mathbf{pk}, R)$;
- **if** $0 \leftarrow \text{NISA.}\textsc{Verify}(\mathsf{param}, \mathbf{pk}, u, P, c)$ **then**
  - return 0;
- return 1;

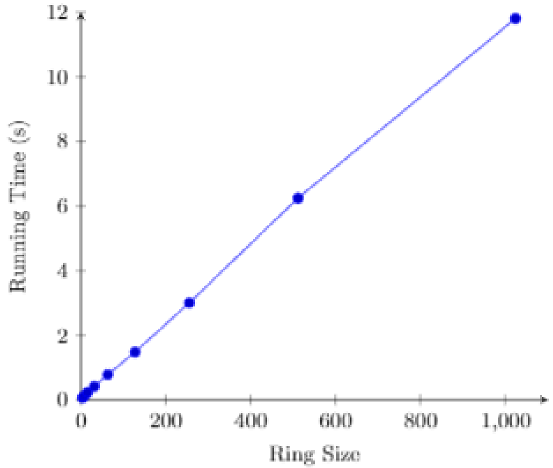
DualRing-EC

- Signature size

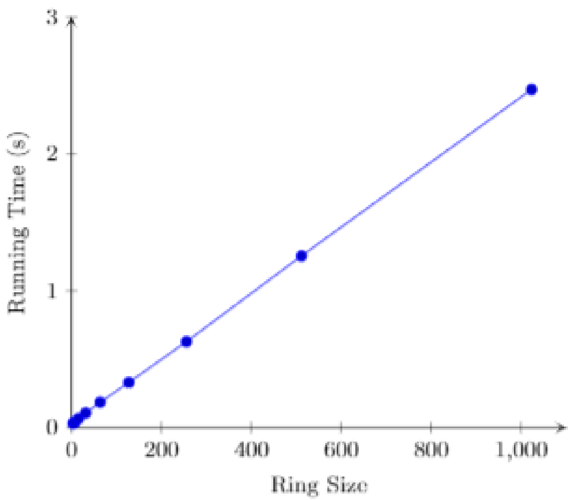


Intel Core i5 2.3GHz, 8GB RAM with MacOS 10.
Python, using the P256 curve in the fastecdsa library

- Running Time



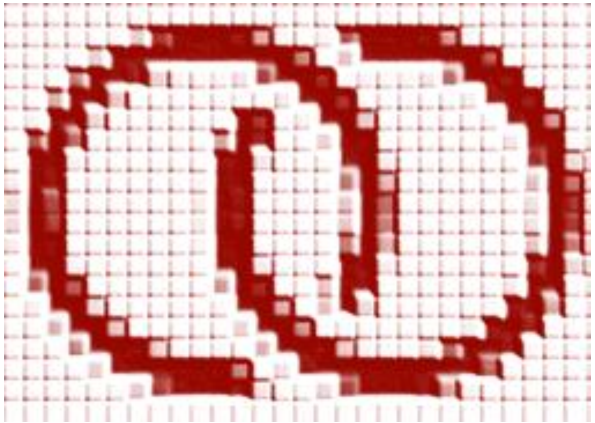
(a) Running time of Sign.



(b) Running time of Verify.

DualRing-LB

- Unlike DualRing-EC, no efficient ZK proof for DualRing in lattice-based setting
- Interestingly, DualRing is already highly efficient in the lattice-based setting:
  - DualRing-LB consists of single response and n challenges.
  - The size of a challenge (around 256 bits) in lattice-based identification is often much smaller than the size of a response (around a few KB).
  - → obtain a compact lattice-based ring signature even without requiring a lattice-based sum argument.



DualRing-LB

- We give a Type-T* canonical identification from M-LWE/SIS, using rejection sampling technique.
- Concrete parameter settings are calculated in the paper.

Ring Signatures	Signature Size (Bytes)						Assumption
	$n=2$	$n=8$	$n=64$	$n=1024$	$n=2048$	$n=4096$	
LAZ@ACNS19	2532	10128	81024	1296384	2592768	6564888576	NTRU
BKP@Asiacrypt20	49000	50000	52000	54000	54500	55000	M-LWE+M-SIS
EZSLL@CCS19	18000	19000	31000	48000	53000	59000	M-LWE+M-SIS
DualRing-LB	4480	4630	6020	31160	55500	106570	M-LWE+M-SIS

O(log n)

O(n)

Shortest for the ring size between 4 - 1946.
→ Useful in practice (e.g., Monero), where the ring size is not very large

	n=32	n=1024	n=32768	
LNS@Crypto21	15960	17270	18730	M-LWE+M-SIS

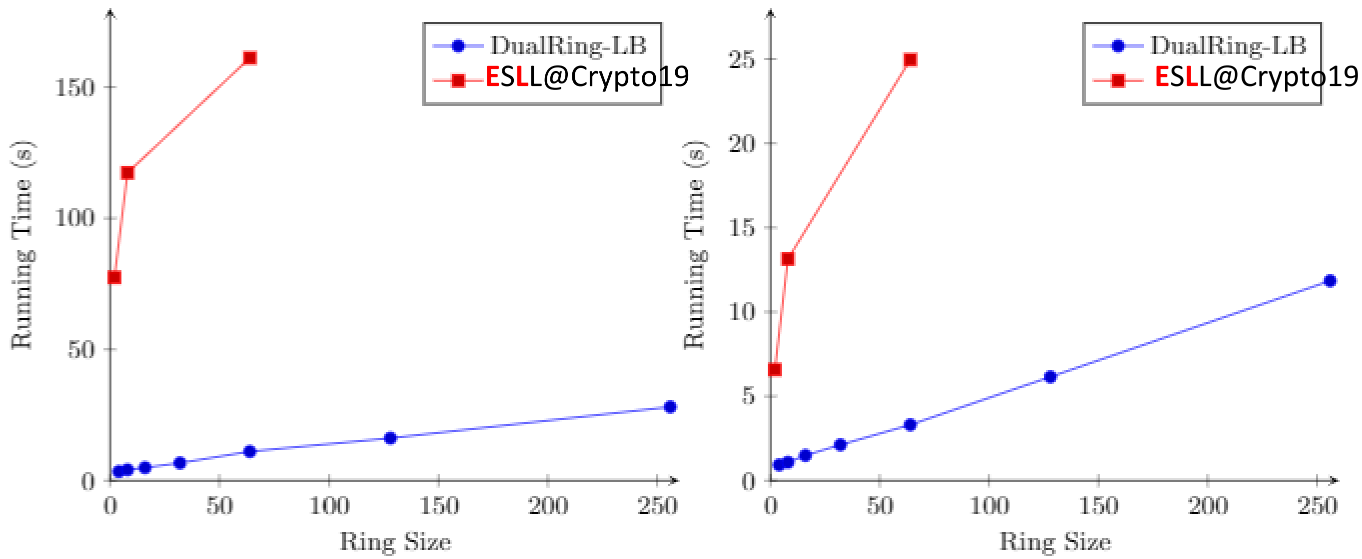
O(log n)

→ DualRing-LB is the shortest for ring size from 4 – 452. (no figure for comparison between 453-505 from [LNS])

[LNS]: Lyubashevsky, Nguyen, Seiler. SMILE: Set Membership from Ideal Lattices with Applications to Ring Signatures and Confidential Transactions. To appear in CRYPTO 2021.

DualRing-LB

- More importantly, much faster in computation
  - Computation in the challenge space is fast



(a) Running Time of Sign.   (b) Running Time of Verify.

Fig. 5: Lattice-based ring signatures

Intel Core i5 2.3GHz, 8GB RAM with MacOS 10.
Python, using the NTT transform in the sympy library

Conclusion

- Propose a generic construction of ring signature scheme using a dual ring structure.
- Instantiated in the DL-setting, it is the shortest ring signature scheme without using trusted setup.
- Instantiated in M-LWE/SIS, we have the shortest ring signature for ring size between 4 and ~500.

Thank you!