KU LEUVEN

Leuven Isogeny Days 3, Sep 21-23

CSI-SharK: CSI-FiSh with Sharing-friendly Keys

Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen

imec-COSIC, KU Leuven, Leuven, Belgium

firstname.lastname@kuleuven.be

ia.cr/2022/1189

Introduction: HHS and ID Protocols

- **Hard Homogeneous Space (HSS) [Cou06]**:
  - A set $\mathcal{E}$ equipped with a group action by a group G.
  - If $G =< g >$ cyclic of order N, then we can define
  - $[a]([b]E) = [a+b]E$

$$\star : G \times \mathcal{E} \to \mathcal{E}$$
$$g \star E \to E'$$
$$[\,] : Z_N \times \mathcal{E} \to \mathcal{E}$$
$$[a]E = g^a \star E$$

Vectorization Problem:
- Given $[a]E$ find a.

  - Symmetric HSS: HSS is symmetric around $E_0 \in \mathcal{E}$ if one can compute $[-a]E_0$ from $[a]E_0$.

- **ID Protocol**: A cryptographic primitive that allows someone's identity to be confirmed.
  - *Correctness*
  - *Special Sound*: given (a, c, r) and (a, c', r'), can extract w.
  - *HVZK*: Honest verifier only learns whether $(x, w) \in R_L$.



Prover (x, w)

a →

← c

r →



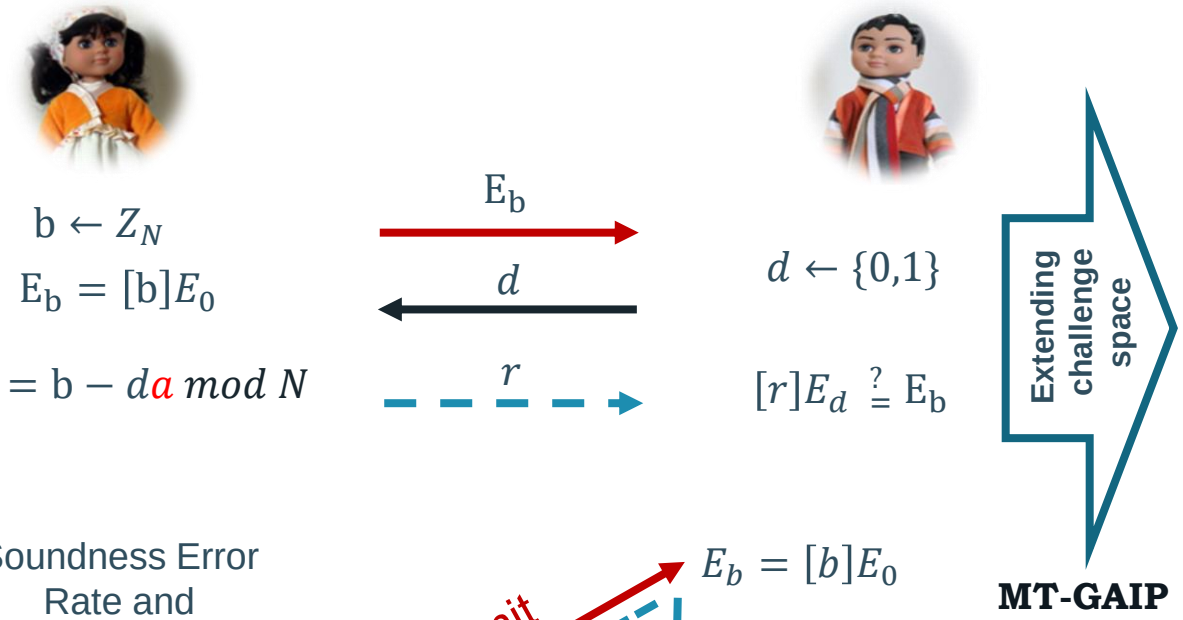
Verifier (x)

- **In the (Q)ROM, using FS, any interactive public-coin HVZK ID protocol**
  - Can be turned into a Non-Interactive ZK ID protocol for the same R_L $\quad c := Hash(x \parallel \text{Com})$
  - Can be turned into a *secure* signature scheme $\quad c := Hash(x \parallel \text{Com} \parallel \text{Message})$

ID Protocols Behind CSI-FiSh (&SeaSign): [Cou06,RS06,DG19,BKV19]

➢ **The ID Protocol for GAIP ($E_1 := [a]E_0$)**

I know a $s.t.$ $E_0, E_1 := [a]E_0$

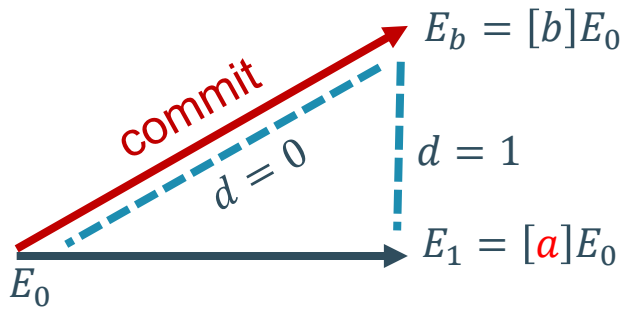


Extending challenge space

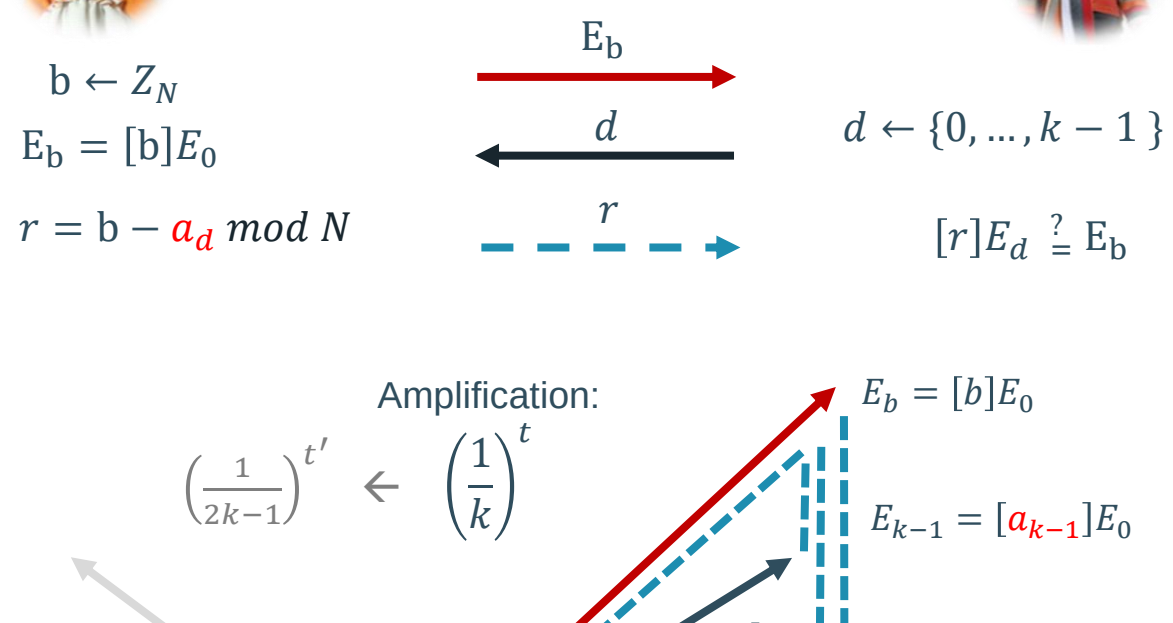
MT-GAIP

Soundness Error Rate and Amplification:

$$\left(\frac{1}{2}\right)^{\lambda}$$



For $E_0, E_1, \cdots, E_{k-1}$, *I know* a $s.t.$ $E_i := [a]E_j$ for some $i, j \in \{0, 1, \ldots, k-1\}$ *and* $i \neq j$



Amplification:

$$\left(\frac{1}{2k-1}\right)^{t'} \leftarrow \left(\frac{1}{k}\right)^{t}$$

CSI-FiSh Signature and Threshold Variants: [BKV19, DM20, CS21]

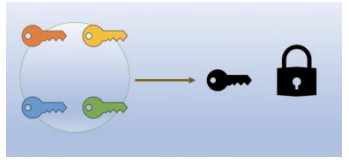
- **Key Generation (KG):** Given a curve E_0,
  - For $i = 1, 2, \dots, k-1$: Sample $a_i \leftarrow Z_N$; Set $E_i = [a_i]E_0$;
  - Return $(sk, pk) := ((a_1, \dots, a_{k-1}), (E_0, E_1, \dots, E_{k-1}))$

- **Signing:** Given m and $sk := (a_1, \dots, a_{k-1})$
  - For $j = 1, 2, \dots, t$:
    - Sample $b_j \leftarrow Z_N$; Set $E_{b_j} = [b_j]E_0$;
  - Set $(d_1 \| d_2 \| \cdots \| d_t) := Hash(m \| E_{b_1} \| \cdots \| E_{b_t})$;
  - For $j = 1, 2, \dots, t$: Set $r_j = b_j - a_{d_j}$;
  - Set $\sigma := ((d_1, d_2, \dots, d_t), (r_1, r_2, \dots, r_t))$;
  - Return (m, σ),

- **Verification:** Given m, σ and $pk := (E_0, E_1, \dots, E_{k-1})$:
  - Parse $\sigma := \big((d_1, d_2, \dots, d_t), (r_1, r_2, \dots, r_t)\big)$
  - For $j = 1, 2, \dots, t$: Set $\hat{E}_{b_j} := [r_j]E_{d_j}$;
  - Set $(\hat{d}_1 \| \hat{d}_2 \| \cdots \| \hat{d}_t) := Hash(m \| \hat{E}_{b_1} \| \cdots \| \hat{E}_{b_t})$;
  - Return 1 if for $j = 1, 2, \dots, t$, $\hat{d}_j = d_j$; otherwise, 0;



- **Threshold Signatures:**



  - *KG:* by a TTP or a DKG protocol
  - *Signing*: by a qualified set of parties
  - *Verification*: by anyone which has pk

a_A 

$E_1 = [a]E_0$

$a = a_A + a_B$

 a_B

$E_{1A} = [a_A]E_0$ $\qquad$ $E_{1B} = [a_B]E_0$

$$([a_A]E_0) \cdot ([a_B]E_0) = ?\,!?\,!?$$

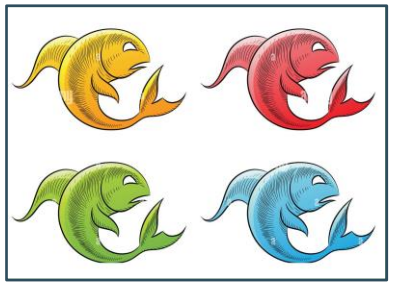
PROBLEM Such an operation does not exist in this setting!

SOLUTION Round Robin Computation!

$$E_{1A} = [a_A]E_0 \Rightarrow E_1 = [a_B]E_{1A} = [a_B + a_A]E_0$$

Threshold Signatures & DKG Based on CSI-FiSh and CSIDH:

- **[De Feo-Meyer, PKC2020]**
  - A passively secure Threshold Signature (TS) based on CSI-FiSh
  - Key generation is done with a TTP
  - Works with Shamir Secret Sharing (SSS)
  - Efficient



- **Sashimi [Cozzo-Smart, PQ2020]**
  - An actively secure (with abort) Threshold Signature based on CSI-FiSh (Uses ZK proofs to achieve active security)
  - Key generation is done with a DKG protocol
  - Works with Replicated Secret Sharing (RSS)
  - Very slow: 2 parties sign a message in about 283 seconds



- **CSI-RAShi [BDPV, PQ2021]:**
  - A robust and actively secure Distributed Key Generation (DKG) protocol for generating $E_1 = [a]E_0$
  - Works with Shamir Secret Sharing (SSS) scheme
  - Can be used to sample the (secret and public) keys of CSI-FiSh signature in the threshold setting
  - Very slow: computing a single curve $[a]E_0$ with n parties takes $4.5 + 18n$ seconds!



On the Efficiency of TS & DKG Protocol: [DM20, CS20, BDPV21]

To sample $(sk, pk) := ((a_1, \dots, a_{k-1}), (E_0, E_1, \dots, E_{k-1}))$ with n parties:

Key Generation of [DM20] (with a TTP): Requires
- k-1 Shamir Secret Sharing
- k-1 Group Actions (GAs) to compute pk
- $(k-1)|a_i|$ communication with each party

Distributed Key Generation by Sashimi [CS20]: Requires
- k-1 Replicated Secret Sharing
- k-1 GAs to commit & k-1 NIZK prove & verify for $E_{com} = [a_i^l]E_0$
- k-1 GAs to compute pk in a **round robin way**
- k-1 NIZK proofs for $(E_{com} = [a_i^l]E_0) \wedge (E_i = [a_i^l]E_i^{l-1})$
- $k\lambda|a_i|$ communication for each party

Distributed Key Generation by CSI-RAShi [BDPV21]: Requires
- k-1 Verifiable SSS (with $2(k-1)$ piecewise verifiable proofs)
- k-1 GAs to commit to their share, $E_{com} = [a_i^l]E_0$
- k-1 GAs to compute pk in a **round robin way**
- k-1 NIZK proofs for $(E_{com} = [a_i^l]E_0) \wedge (E_i = [a_i^l]E_i^{l-1})$
- $k\lambda(3n + |a_i|)$ communication for each party







To sign a message m, with n parties:

Using the signing protocol of [DM20], a party l needs
- t GAs to compute $[b_j^l]E_0$ in a **round robin way**

Using the signing protocol of [CS20], a party l needs
- t GAs to commit & t NIZK prove & verify for $E_{com}^{j,l} = [b_j^l]E_0$
- t GAs to compute $[b_j]E_0$ in a **round robin way**
- t NIZK proofs for $(E_{com}^{j,l} = [b_j^l]E_0) \wedge (E_j^l = [b_j^l]E_j^{l-1})$

Common values for k and t in practice:

✓ $(k = 2^{12},\ t = 9) \Rightarrow$ KG can be $450\times$ slower than sign!
✓ $(k = 2^{15},\ t = 7) \Rightarrow$ KG can be $4500\times$ slower than sign!

Key Generation of CSI-FiSh is very slow in the threshold setting!



Our Results:

CSI-SharK: CSI-FiSh with Sharing-friendly Keys

☺ as efficient as CSI-FiSh, ☺ single secret key, 😐 relies on a new assumption [BCP21]

We show that Sashimi [CS20] is not secure in the threshold case, but can be used as a full-threshold scheme.

Using CSI-SharK, we present more efficient threshold signatures and a DKG protocol.

Some strategies for parallel computations in isogeny-based schemes:
- Using multi-core CPUs
- Parallel round-robins
- Subdividing a long ZK proof into small ones

Structured Public Keys & the ID Protocol Behind CSI-SharK:

- **Structured Public Keys (SPK) [BCP21]**

For particular k, and N with smallest prime factor $q \geq k+1$.

$$\mathbf{SPK} \coloneqq \{E_0, E_1 \coloneqq [a]E_0, E_2 \coloneqq [c_2 a]E_0, \cdots, E_k \coloneqq [c_k a]E_0\}$$

where $\mathbb{C}_k = \{c_0 = 0, c_1 = 1, c_2, \cdots, c_k\}$, is an exceptional set. e.g., $\mathbb{C}_k = \{c_0 = 0, c_1 = 1, c_2 = 2, \cdots, c_k = k\}$.

Relaying on a TTP, a SPK allows to extend the challenge space of the basic ID protocol to prove knowledge of a in $E_1 \coloneqq [a]E_0$.

- **Security of SPK rely on a new assumption, $\mathbb{C}_k$-VPwAI** (Vectorization Problem with Auxiliary Inputs)

Let $c_0 = 0, c_1 = 1, c_2, \cdots, c_k$ be public elements from (super)exceptional set $\mathbb{C}_k$ with $k < q$ (or $k < q/2$). Given $\mathbb{C}_k \coloneqq \{c_0 = 0, c_1 = 1, c_2, \cdots, c_k\}$, and $E_0, E_1 \coloneqq [c_1 a]E_0, E_2 \coloneqq [c_2 a]E_0, \cdots, E_k \coloneqq [c_k a]E_0$, finding a is hard.

For $E_0, E_1, \cdots, E_{k-1}$, and $\mathbb{C}_k$, *I know* a' $s.t.$ $E_i \coloneqq [a']E_j$ for some $i, j \in \{0, 1, \ldots, k-1\}$ *and* $i \neq j$

Prover



Verifier



$b \leftarrow Z_N$

$E_b = [b]E_0$

$r = b - da \bmod N$

E_b →

← d

r - - - →

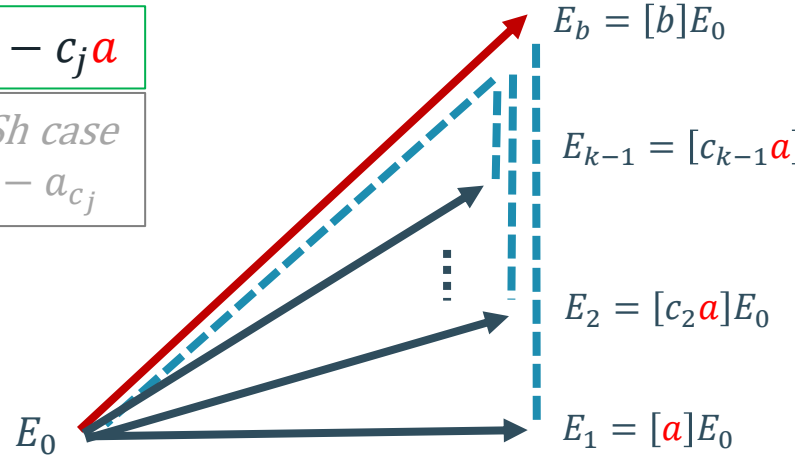
$d \leftarrow \{0, \ldots, k-1\}$

$[r]E_d \stackrel{?}{=} E_b$

$a' = c_i a - c_j a$

in CSI-FiSh case $a' = a_{c_i} - a_{c_j}$

Soundness Amplification: $\left(\frac{1}{k}\right)^t$



CSI-SharK Signature & New Threshold Protocols:

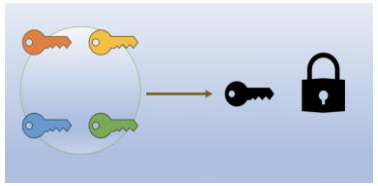
- **Key Generation (KG):** Given a curve E_0, $\mathbb{C}_k = \{0, 1, \cdots, k\}$:
  - For $i = 1, 2, \ldots, k-1$: Sample $a \leftarrow Z_N$; Set $E_i = [ia]E_0$;
  - Return $(sk, pk) := (a, (E_0, E_1, \ldots, E_{k-1}))$

- **Signing:** Given m and $sk = a$
  - For $j = 1, 2, \ldots, t$:
    - Sample $b_j \leftarrow Z_N$; Set $E_{b_j} = [b_j]E_0$;
  - Set $(d_1 \| d_2 \| \cdots \| d_t) := Hash(m \| E_{b_1} \| \cdots \| E_{b_t})$;
  - For $j = 1, 2, \ldots, t$: Set $r_j = b_j - d_j a$;
  - Set $\sigma := ((d_1, d_2, \ldots, d_t), (r_1, r_2, \ldots, r_t))$;
  - Return (m, σ),

- **Verification:** Given m, σ and $pk := (E_0, E_1, \ldots, E_{k-1})$:
  - Parse $\sigma := ((d_1, d_2, \ldots, d_t), (r_1, r_2, \ldots, r_t))$
  - For $j = 1, 2, \ldots, t$: Set $\hat{E}_{b_j} := [r_j]E_{d_j}$;
  - Set $(\hat{d}_1 \| \hat{d}_2 \| \cdots \| \hat{d}_t) := Hash(m \| \hat{E}_{b_1} \| \cdots \| \hat{E}_{b_t})$;
  - Return 1 if for $j = 1, 2, \ldots, t$, $\hat{d}_j = d_j$; otherwise, 0;



Threshold Signatures



Distributed Key Generation



☺ Single Secret Key → Single Secret Sharing

☺ Single Secret Sharing → Single Commitment to a, and a single NIZK proof for $E_{com} = [a]E_0$

☺ Single Secret Sharing → $|a|$ Communication (instead of $(k-1)|a_i|$)

☺ Structured Public Key → Single ZK proof for

$$(E^l_{com} = [b_l]E_0) \wedge (E^l_i = [c_1 b_l]E^{l-1}_i \wedge \ \ldots \wedge E^l_i = [c_k b_l]E^{l-1}_i)$$

instead of $k-1$ ZK proofs for

$$(E^l_{com} = [b_{l,i}]E_0) \wedge (E^l_i = [b_{l,i}]E^{l-1}_i)$$

We construct a Σ-protocol for the new language using conjunction of two sigma protocols.

CSI-FiSh-based vs. CSI-SharK-based DKG Protocols:

CSI-FiSh-based

To sample $(\boldsymbol{sk}, \boldsymbol{pk}) \coloneqq ((\boldsymbol{a_1}, \dots, \boldsymbol{a_{k-1}}), (\boldsymbol{E_0}, \boldsymbol{E_1}, \dots, \boldsymbol{E_{k-1}}))$

Key Generation of [DM20] (with a TTP): Requires
- k-1 Shamir Secret Sharing
- k-1 Group Actions (GAs) to compute pk
- $k|a^l|$ communication with each party



Distributed Key Generation by Sashimi [CS20]: Requires
- k-1 ~~Replicated~~ Full-Threshold Secret Sharing
- k-1 GAs to commit & k-1 NIZK prove & verify for $E_{com} = [a_i^l]E_0$
- k-1 GAs to compute pk in a **round robin way**
- k-1 NIZK proofs for $(E_{com} = [a_i^l]E_0) \wedge (E_i = [a_i^l]E_i^{l-1})$
- $k\lambda|a^l|$ communication for each party



Distributed Key Generation by CSI-RAShi [BDPV21]: Requires
- k-1 Verifiable SSS (with $2(k-1)$ piecewise verifiable proofs)
- k-1 GAs to commit to their share, $E_{com} = [a_i^l]E_0$
- k-1 GAs to compute pk in a **round robin way**
- k-1 NIZK proofs for $(E_{com} = [a_i^l]E_0) \wedge (E_i = [a_i^l]E_i^{l-1})$
- $k\lambda(3n + |a^l|)$ communication for each party



CSI-SharK-based

To sample $(\boldsymbol{sk}, \boldsymbol{pk}) \coloneqq (\boldsymbol{a}, (\boldsymbol{E_0}, \boldsymbol{E_1}, \dots, \boldsymbol{E_{k-1}}))$

Key Generation of [DM20] (with a TTP): Requires
- 1 Shamir Secret Sharing
- k-1 Group Actions (GAs) to compute pk
- $|a^l|$ communication with each party

Distributed Key Generation by Sashimi [CS20]: Requires
- 1 Full-Threshold Secret Sharing
- 1 GA to commit & 1 NIZK prove & verify for $E_{com} = [a^l]E_0$
- k-1 GAs to compute pk in a **round robin way**
- 1 NIZK for $(E_{com}^l = [a^l]E_0) \wedge (E_i^l = [c_1 a^l]E_i^{l-1} \wedge \dots \wedge E_i^l = [c_k a^l]E_i^{l-1})$
- $(k + \frac{1}{2}\sqrt{k\lambda})|a^l|$ communication for each party

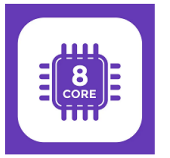
Distributed Key Generation by CSI-RAShi [BDPV21]: Requires
- 1 Verifiable SSS (with 2 piecewise verifiable proofs)
- 1 GAs to commit to their share, $E_{com} = [a_i^l]E_0$
- k-1 GAs to compute pk in a **round robin way**
- 1 NIZK for $(E_{com}^l = [a^l]E_0) \wedge (E_i^l = [c_1 a^l]E_i^{l-1} \wedge \dots \wedge E_i^l = [c_k a^l]E_i^{l-1})$
- $(k + \frac{1}{2}\sqrt{k\lambda})|a^l|$ communication for each party

Parallelization in Isogeny-based Protocols: Using Multi-core CPUs

- **Need to compute multiple independent curves**

$$E_1 = [a_1]E_0,\ E_2 = [a_2]E_0,\ \dots,\ E_k = [a_k]E_0$$

Given multi-core CPUs, each geoup action can be assigned to one core (or thread) of the CPU.



CSI-FiSh [BKV19]

S	t	k	|sk|	|pk|	|sig|	KeyGen	Sign	Verify
2^1	56	16	16 B	128 B	1880 B	100 ms	2.92 s	2.92 s
2^2	38	14	16 B	256 B	1286 B	200 ms	1.98 s	1.97 s
2^3	28	16	16 B	512 B	956 B	400 ms	1.48 s	1.48 s
2^4	23	13	16 B	1 KB	791 B	810 ms	1.20 s	1.19 s
2^6	16	16	16 B	4 KB	560 B	3.3 s	862 ms	859 ms
2^8	13	11	16 B	16 KB	461 B	13 s	671 ms	670 ms
2^{10}	11	7	16 B	64 KB	395 B	52 s	569 ms	567 ms
2^{12}	9	11	16 B	256 KB	329 B	3.5 m	471 ms	469 ms
2^{15}	7	16	16 B	2 MB	263 B	28 m	395 ms	393 ms

CSI-SharK

#Cores	$k-1$	t_S	h	|sk|	|pk|	|sig|	KGen	Sign	Verify
1	2^0	71	16	16 B	64 B	2307 B	40 ms	2840 ms	2840 ms
	2^4	23	15	16 B	1 KB	759 B	640 ms	920 ms	920 ms
	2^8	13	12	16 B	16 KB	436 B	10 s	520 ms	520 ms
	2^{12}	9	11	16 B	256 KB	306 B	164 s	360 ms	360 ms
8	2^0	71	16	16 B	64 B	2307 B	40 ms	360 ms	360 ms
	2^4	23	15	16 B	1 KB	759 B	80 ms	120 ms	120 ms
	2^8	13	12	16 B	16 KB	436 B	1280 ms	80 ms	80 ms
	2^{12}	9	11	16 B	256 KB	306 B	20 s	80 ms	80 ms
16	2^0	71	16	16 B	64 B	2307 B	40 ms	200 ms	200 ms
	2^4	23	15	16 B	1 KB	759 B	40 ms	80 ms	80 ms
	2^8	13	12	16 B	16 KB	436 B	640 ms	40 ms	40 ms
	2^{12}	9	11	16 B	256 KB	306 B	10 s	40 ms	40 ms

This is a strong incentive for using isogeny-based signatures in practice.

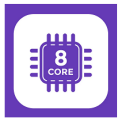
Parallelization in Threshold Schemes: Parallel Round-Robins, Multi-core CPUs, Subdivided ZK Proofs

- **In threshold protocols, we need to compute the same with multiple parties in a round robin way and give a ZK proof.**

$$E_1 = [a_1]E_0,\ E_2 = [a_2]E_0,\ \dots,\ E_k = [a_k]E_0$$

- ✓ **Using parallel round-robin communications and multi-core CPUs, results in considerably more efficient threshold protocols.**





$$\begin{matrix}(\boldsymbol{E_1^A} = [a_1^A]E_0, \pi_1^A), \\ (\boldsymbol{E_1^B} = [a_1^B]\boldsymbol{E_1^A}, \pi_1^B), \\ (E_1^C = [a_1^C]\boldsymbol{E_1^B}, \pi_1^C),\end{matrix} \quad \begin{matrix}(E_2^A = [a_2^A]E_2^B, \pi_2^A), \\ (E_2^B = [a_2^B]E_2^C, \pi_2^B), \\ (E_2^C = [a_2^C]E_0, \pi_2^C),\end{matrix} \quad \dots, \quad \begin{matrix}(E_k^A = [a_k^A]E_k^C, \pi_k^A) \\ (E_k^B = [a_k^B]E_0, \pi_k^B) \\ (E_k^C = [a_k^C]E_k^B, \pi_k^C)\end{matrix}$$

$$P_A \rightarrow P_B \rightarrow P_C \qquad P_C \rightarrow P_B \rightarrow P_A \qquad P_B \rightarrow P_C \rightarrow P_A$$

- **In the CSI-SharK-based protocols the parties need to give a ZK proof for k curves, which can be inefficient for a large value of k.**



- ✓ **We subdivide the one full proof of $k-1$ curves into smaller proofs.**

  e.g., $\pi := \pi_1 + \pi_2$

$$(E_{com} = [a]E_0) \wedge (E_1 = [c_1 a]E_0 \wedge \dots \wedge \mathrm{E_k} = [c_k a]E_0) \begin{cases} (E_{com} = [a]E_0) \wedge (E_1 = [c_1 a]E_0 \wedge \dots \wedge \mathrm{E_{k/2}} = [c_{k/2} a]E_0) \\ (E_{com} = [a]E_0) \wedge (E_{\frac{k}{2}+1} = [c_{\frac{k}{2}+1} a]E_0 \wedge \dots \wedge \mathrm{E_k} = [c_k a]E_0) \end{cases}$$

Performance of Threshold Signatures & DKG Protocols:

- In both non-threshold, and threshold cases by using multi-core CPUs, we can gain a lot!
- Communication between parties, & the secret key size in SharK-based protocols are considerably smaller.
- Signing protocols have the same efficiency. However, using our optimizations, via multi-core CPUs, 3 parties can sign in 28 seconds. For the same setting, with CSI-SharK, the DKG with 3 parties can be done in 4 min, just $8\times$ slower than Signing.
- We have similar improvements in the DKG protocol CSI-RAShi. Using our protocol, we estimate 8 parties can generate a 1 KB public key (i.e., $k = 2^4$), in about 65 seconds.
- **Our results show that isogeny-based non-threshold and threshold signatures can be practical on current devices with multi-core CPUs.**
- More discussions and results (details of the flaw in Sashimi) in the paper.

Full-Threshold Case	$k-1$	CSI-FiSh naive [12]	CSI-FiSh staggered (Sec. 4)	CSI-FiSh comm. (Sec. 4)	CSI-SharK naive (Sec. 5)	CSI-SharK staggered (Sec. 5)	Signing[8] (App. A)
3 parties	2^4	17 min	10 min	131 kB	7.3 min	9.1 kB / 5.5 min / 21 kB	12 min
each with	2^8	4.5 h	2.6 h	2.0 MB	1.9 h	24 kB / 65 min / 84 kB	7.2 min
1 core	2^{12}	73 h	42 h	33 MB	31 h	264 kB / 16 h / 517 kB	4.5 min
3 parties	2^4	67 s	38 s		27 s	21 s	47 s
each with	2^8	18 min	10 min		7 min	4 min	28 s
16 cores	2^{12}	4.8 h	2.6 h		1.9 h	60 min	18 s
8 parties	2^4	3.1 min	1.8 min		80 s	60 s	147 s
each with	2^8	50 min	28 min		21 min	12 min	94 s
16 cores	2^{12}	13 h	7.4 h		5.6 h	2.9 h	87 s

Shamir secret sharing	$k-1$	CSI-FiSh naive [2]	CSI-FiSh staggered (Sec. 4)	CSI-FiSh comm. (Sec. 4)	|sk|	CSI-SharK naive (Sec. 5)	CSI-SharK staggered (Sec. 5)	|sk|
3 parties	2^4	18 min	13 min	137 kB	512 B	8.5 min / 9.5 kB	6.5 min / 26 kB	32 B
each with	2^8	4.7 h	3.3 h	2.1 MB	8 kB	2.2 h / 25 kB	65 min / 89 kB	32 B
1 core	2^{12}	76 h	53 h	34 MB	128 kB	35 h / 265 kB	18 h / 522 kB	32 B
3 parties	2^4	69 s	48 s			32 s	24 s	
each with	2^8	18 min	13 min			8.2 min	4.7 min	
16 cores	2^{12}	4.9 h	3.3 h			2.2 h	68 min	
8 parties	2^4	2.9 min	2.1 min	146 kB		85 s / 10 kB	65 s / 26 kB	
each with	2^8	47 min	33 min	2.3 MB		22 min / 25 kB	12 min / 89 kB	
16 cores	2^{12}	12 h	8.8 h	36 MB		5.8 h / 265 kB	3.0 h / 522 kB	

KU LEUVEN

Thank You!



shahla.atapoor@kuleuven.be
karim.baghery@kuleuven.be
daniele.cozzo@kuleuven.be
robi.pedersen@kuleuven.be