Securing DNSSEC Keys via Threshold ECDSA From Generic MPC

Kris Shrishak

TU Darmstadt, Germany

November 6, 2020
NIST Workshop on Multi-Party Threshold Schemes 2020

Based on work published at ESORICS'20
with Anders Dalskov, Marcel Keller, Claudio Orlandi and Haya Shulman

This work

Threshold ECDSA for DNS zone signing

This work

Threshold ECDSA for DNS zone signing

- Key security for DNSSEC
- Generic way of doing threshold ECDSA (signing and key gen)
- Support for lots of different threat models
- As fast, or faster, than previous work

Outline

Outline

DNS and DNSSEC

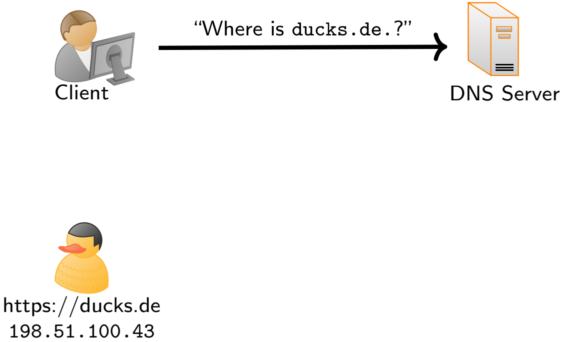
Threshold signatures for DNSSEC

DNS

DNS is a protocol for mapping names to addresses

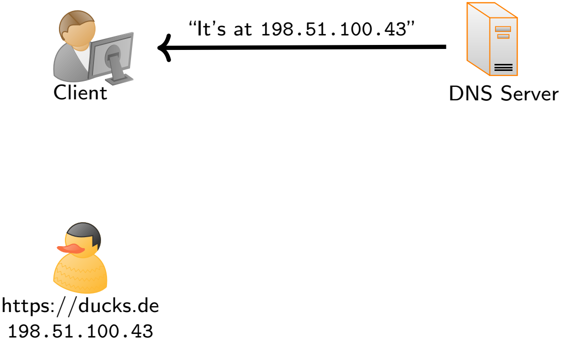
DNS

DNS is a protocol for mapping names to addresses



DNS

DNS is a protocol for mapping names to addresses



DNS

DNS is a protocol for mapping names to addresses

DNS Insecurity

Poisoning/Spoofing is possible

DNS Insecurity

Poisoning/Spoofing is possible

First answer is accepted

DNS Insecurity

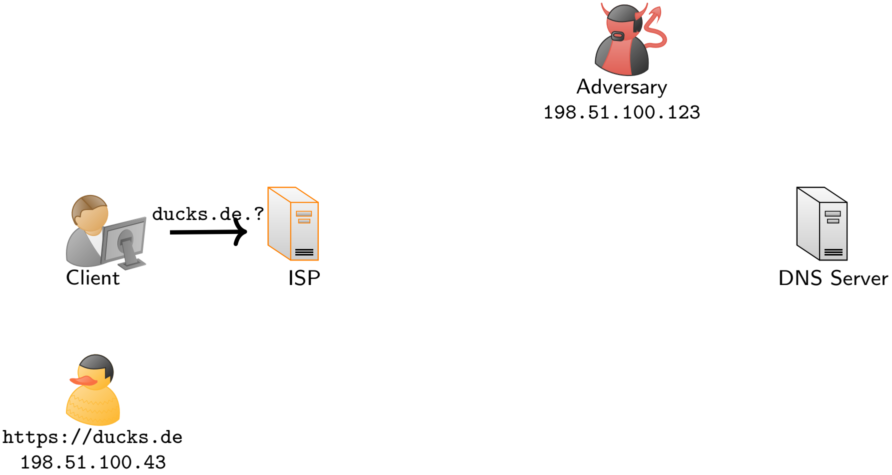
Poisoning/Spoofing is possible

First answer is accepted

DNS Insecurity

Poisoning/Spoofing is possible

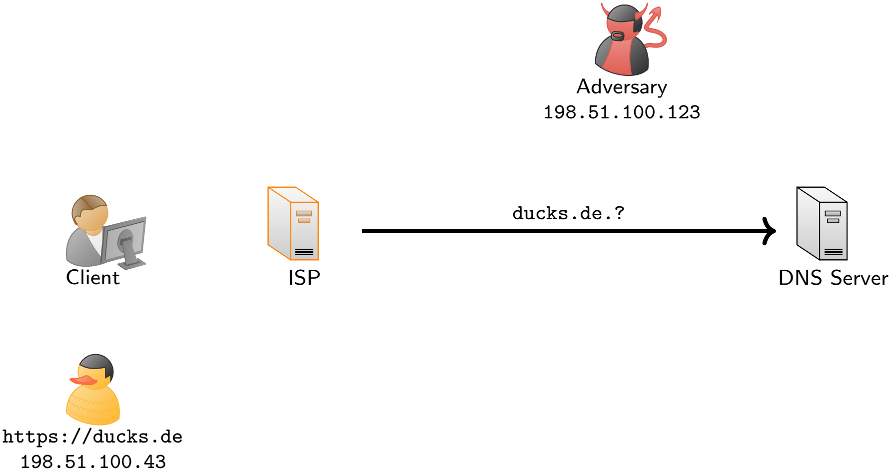
First answer is accepted



DNS Insecurity

Poisoning/Spoofing is possible

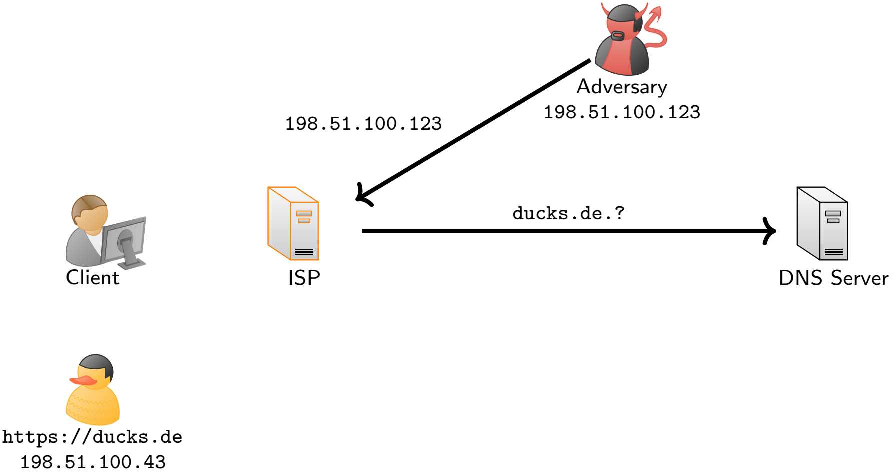
First answer is accepted



DNS Insecurity

Poisoning/Spoofing is possible

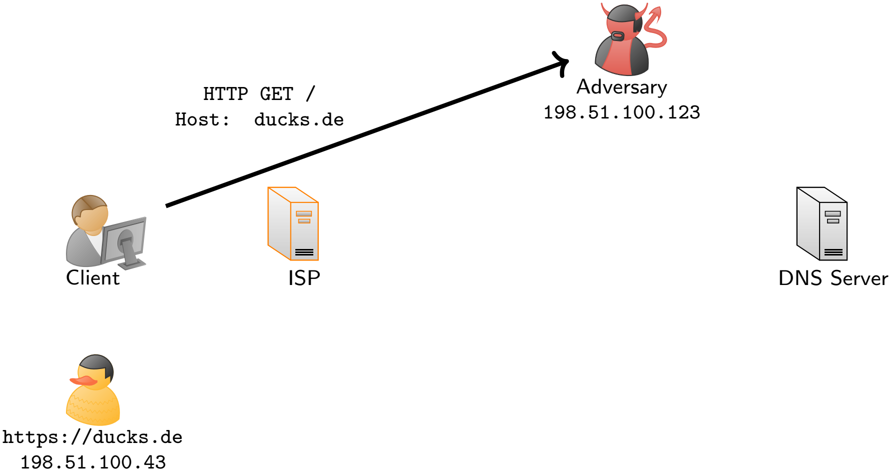
First answer is accepted



DNS Insecurity

Poisoning/Spoofing is possible

First answer is accepted

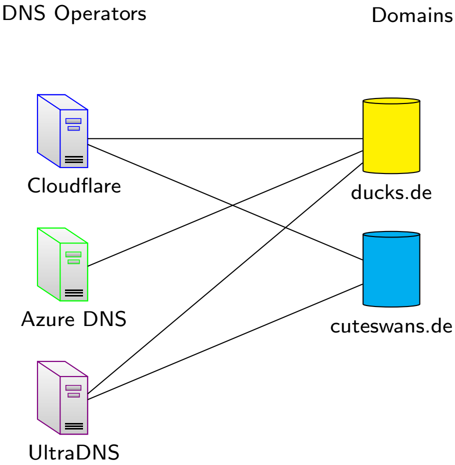


DNSSEC

DNSSEC fixes this problem

- Data integrity: data was not changed in transit
- Origin authentication: data originated from the owner

DNS in practice



DNSSEC deployment issues

Studies [1,2] have found that

- Some operators use the same key for all domains
  - E.g., one key shared by $> 132\,000$ domains

[1] A Longitudinal, End-to-End View of the DNSSEC Ecosystem (USENIX '17)
[2] One Key to Sign Them All Considered Vulnurable: Evaluation of DNSSEC in the Internet (NSDI '17)

DNSSEC deployment issues

Studies [1,2] have found that

- Some operators use the same key for all domains
  - E.g., one key shared by $> 132\,000$ domains

[1] A Longitudinal, End-to-End View of the DNSSEC Ecosystem (USENIX ’17)

[2] One Key to Sign Them All Considered Vulnurable: Evaluation of DNSSEC in the Internet (NSDI ’17)

DNSSEC deployment issues

Studies [1,2] have found that

- Some operators use the same key for all domains
  - E.g., one key shared by $> 132\,000$ domains
- Default is 1024-bit RSA
  - Most keys 1024-bit, with $\sim$10K domains use 512-bit RSA
  - The majority of keys were not rotated in a 21-month period
  - Some providers use different keys but share the modulus

[1] A Longitudinal, End-to-End View of the DNSSEC Ecosystem (USENIX '17)

[2] One Key to Sign Them All Considered Vulnurable: Evaluation of DNSSEC in the Internet (NSDI '17)

DNSSEC in practice

DNSSEC

- Should use ECDSA instead of RSA
  - Shorter signatures reduce the chance of packet fragmentation [1]

[1] RFC 6781 recommends 1024-bit RSA for this reason
[2] See 2016 Dyn attacks
[3] RFC 8901: Multi-Signer DNSSEC Models

DNSSEC in practice

DNSSEC

- Should use ECDSA instead of RSA
  - Shorter signatures reduce the chance of packet fragmentation [1]

- Support multiple name servers
  - better availability and DDoS protection [2]
  - new standard [3] requires zone owner interaction while relinquishing key control

[1] RFC 6781 recommends 1024-bit RSA for this reason

[2] See 2016 Dyn attacks

[3] RFC 8901: Multi-Signer DNSSEC Models

Outline

DNS and DNSSEC

Threshold signatures for DNSSEC

Threshold signatures for DNSSEC

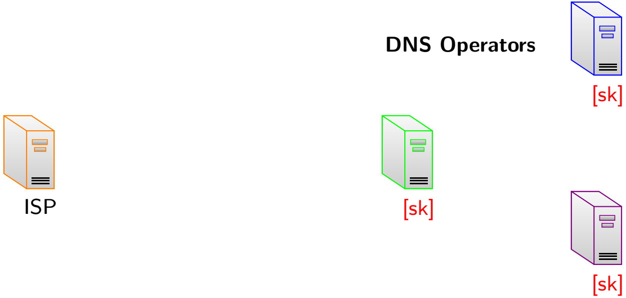
Zone signing with Threshold ECDSA

$[\mathrm{sk}] \leftarrow \mathit{Share}(\mathrm{sk})$

Threshold signatures for DNSSEC

Zone signing with Threshold ECDSA

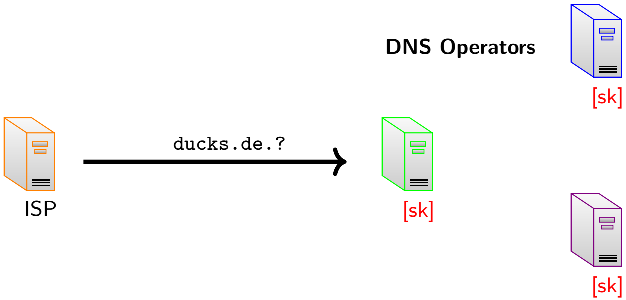
$[\text{sk}] \leftarrow \mathit{Share}(\text{sk})$



Threshold signatures for DNSSEC

Zone signing with Threshold ECDSA

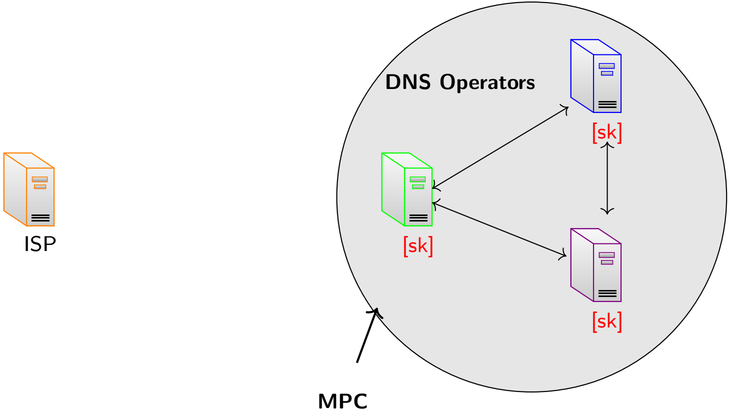
$[sk] \leftarrow Share(sk)$



Threshold signatures for DNSSEC

Zone signing with Threshold ECDSA

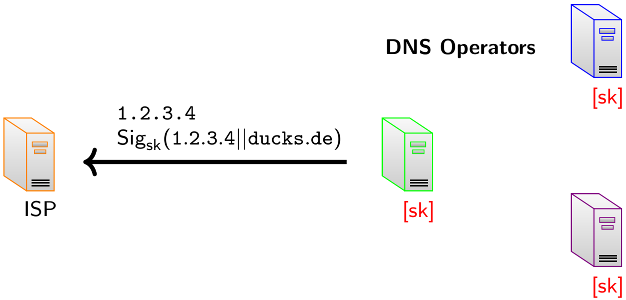
$[\mathsf{sk}] \leftarrow \mathit{Share}(\mathsf{sk})$



Threshold signatures for DNSSEC

Zone signing with Threshold ECDSA

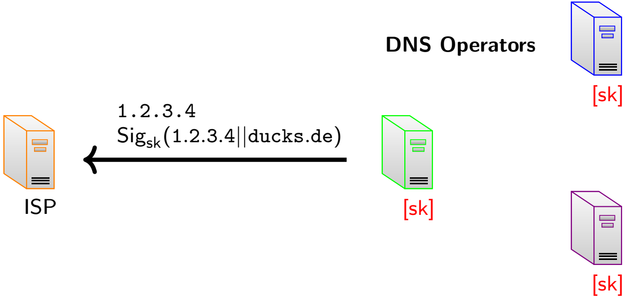
$[\mathsf{sk}] \leftarrow \mathit{Share}(\mathsf{sk})$



Threshold signatures for DNSSEC

Zone signing with Threshold ECDSA

$[\text{sk}] \leftarrow \textit{Share}(\text{sk})$



Threshold signing should not be much more expensive than regular DNSSEC

ECDSA

$$s = k^{-1}(H(M) + \mathsf{sk} \cdot r_x)$$

ECDSA

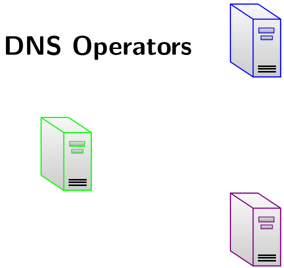
$$s = k^{-1}(H(M) + \mathsf{sk} \cdot r_x)$$

Threshold ECDSA

$$s = H(M)[k^{-1}] + [\mathsf{sk} \cdot k^{-1}] \cdot r_x$$

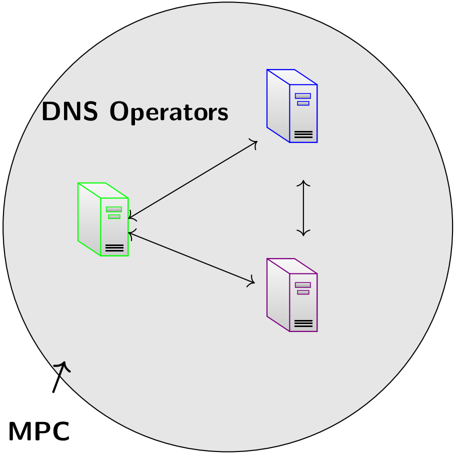
Threshold ECDSA signing in 3 phases

$$s = H(M)[k^{-1}] + [\mathsf{sk} \cdot k^{-1}] \cdot r_x$$



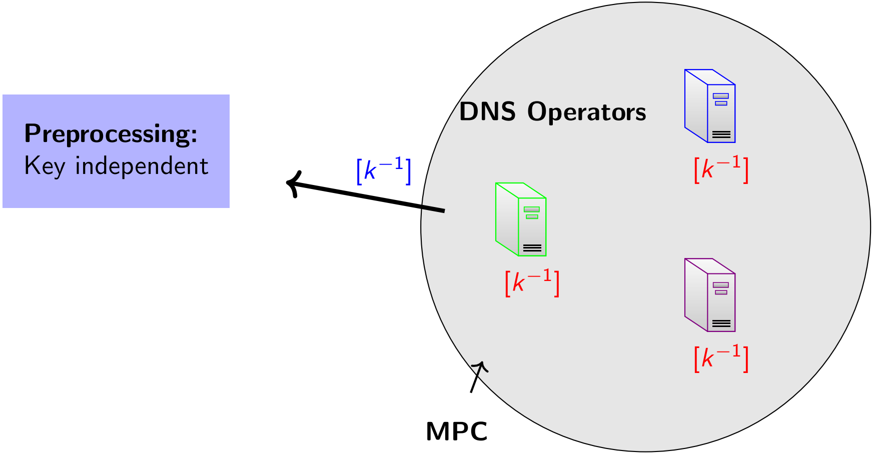
Threshold ECDSA signing in 3 phases

$$s = H(M)[k^{-1}] + [\mathsf{sk} \cdot k^{-1}] \cdot r_x$$



Threshold ECDSA signing in 3 phases

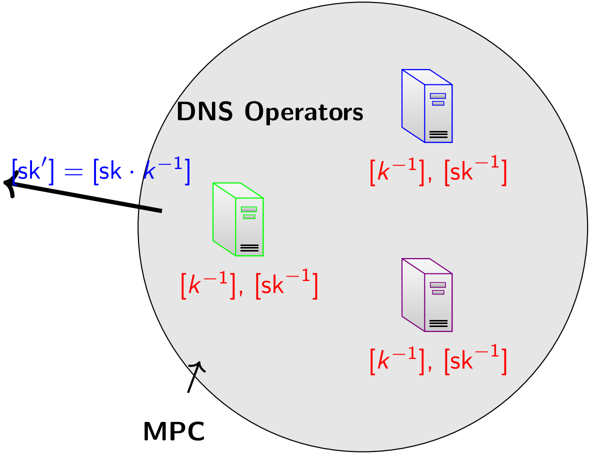
$$s = H(M)[k^{-1}] + [\mathsf{sk} \cdot k^{-1}] \cdot r_x$$



Threshold ECDSA signing in 3 phases

$$s = H(M)[k^{-1}] + [\mathsf{sk} \cdot k^{-1}] \cdot r_x$$

Preprocessing:
Key independent
Message independent

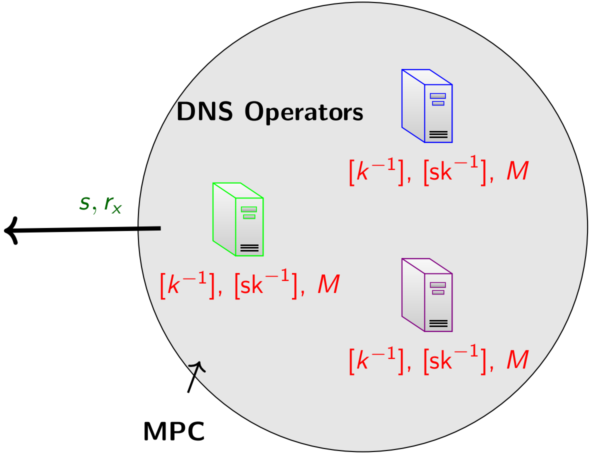


Threshold ECDSA signing in 3 phases

$$s = H(M)[k^{-1}] + [\mathrm{sk} \cdot k^{-1}] \cdot r_x$$

Preprocessing:
Key independent
Message independent

Online phase



Threshold ECDSA signing

$$s = H(M)[k^{-1}] + [\mathsf{sk} \cdot k^{-1}] \cdot r_x$$

Problems: How do we compute

1. $[k^{-1}]$
2. r_x

Threshold ECDSA signing

Need to compute $s = [k^{-1}](H(M) + [\mathsf{sk}] \cdot r_x)$

Threshold ECDSA signing

Need to compute $s = [k^{-1}](H(M) + [\mathsf{sk}] \cdot r_x)$

Problem how do we compute $[k^{-1}]$?

Main difficulty with threshold ECDSA

Threshold ECDSA signing

From $[k]$ to $[k^{-1}]$ using a trick due to Bar-Ilan and Beaver[4]

[4] Non-cryptographic fault-tolerant computing in constant number of rounds of interaction (PODC '89)

Threshold ECDSA signing

From $[k]$ to $[k^{-1}]$ using a trick due to Bar-Ilan and Beaver[4]

1. Suppose we have $([k], [b], [c])$ with $c = k \cdot b$

[4] Non-cryptographic fault-tolerant computing in constant number of rounds of interaction (PODC '89)

Threshold ECDSA signing

From $[k]$ to $[k^{-1}]$ using a trick due to Bar-Ilan and Beaver[4]

1. Suppose we have $([k], [b], [c])$ with $c = k \cdot b$
2. Open $[c]$

[4] Non-cryptographic fault-tolerant computing in constant number of rounds of interaction (PODC '89)

Threshold ECDSA signing

From $[k]$ to $[k^{-1}]$ using a trick due to Bar-Ilan and Beaver[4]

1. Suppose we have $([k], [b], [c])$ with $c = k \cdot b$
2. Open $[c]$
3. Compute $c^{-1}[b] = [(k \cdot b)^{-1} b] = [k^{-1}]$

[4] Non-cryptographic fault-tolerant computing in constant number of rounds of interaction (PODC '89)

Threshold ECDSA signing

From $[k]$ to $[k^{-1}]$ using a trick due to Bar-Ilan and Beaver[4]

1. Suppose we have $([k], [b], [c])$ with $c = k \cdot b$
2. Open $[c]$
3. Compute $c^{-1}[b] = [(k \cdot b)^{-1} b] = [k^{-1}]$

Computing $[k^{-1}]$ is the most expensive part of signing

---

[4] Non-cryptographic fault-tolerant computing in constant number of rounds of interaction (PODC '89)

Secure Computation over Elliptic Curves

Need to compute $s = [k^{-1}](H(M) + [\mathsf{sk}] \cdot r_x)$

Secure Computation over Elliptic Curves

Need to compute $s = [k^{-1}](H(M) + [\mathsf{sk}] \cdot r_x)$

Problem how do we compute r_x?

Secure Computation over Elliptic Curves

Need to compute $s = [k^{-1}](H(M) + [\mathsf{sk}] \cdot r_x)$

Problem how do we compute r_x?

where

$$(r_x, r_y) = R = k \cdot G$$

Secure Computation over Elliptic Curves

Need to compute $s = [k^{-1}](H(M) + [\mathsf{sk}] \cdot r_x)$

Problem how do we compute r_x?

where

$$(r_x, r_y) = R = k \cdot G$$

Secure Computation over Elliptic Curves

Let $[k]$ denote an additive sharing of k over $\mathbb{Z}_p$

Let $\langle k \rangle$ denote a sharing of $k \cdot G$.

Secure Computation over Elliptic Curves

Let $[k]$ denote an additive sharing of k over $\mathbb{Z}_p$

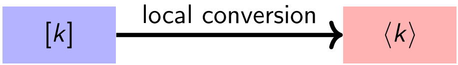
Let $\langle k \rangle$ denote a sharing of $k \cdot G$.

$[k]$ $\langle k \rangle$

Secure Computation over Elliptic Curves

Let $[k]$ denote an additive sharing of k over $\mathbb{Z}_p$

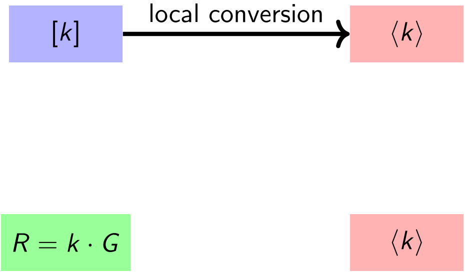
Let $\langle k \rangle$ denote a sharing of $k \cdot G$.



Secure Computation over Elliptic Curves

Let $[k]$ denote an additive sharing of k over $\mathbb{Z}_p$

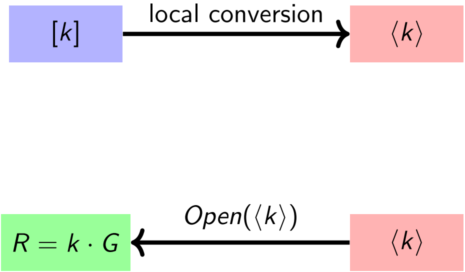
Let $\langle k \rangle$ denote a sharing of $k \cdot G$.



Secure Computation over Elliptic Curves

Let $[k]$ denote an additive sharing of k over $\mathbb{Z}_p$

Let $\langle k \rangle$ denote a sharing of $k \cdot G$.



Secure Computation over Elliptic Curves

Let $[k]$ denote an additive sharing of k over $\mathbb{Z}_p$

Let $\langle k \rangle$ denote a sharing of $k \cdot G$.

Supports all the usual suspects

- Addition/constant addition
- Constant scalar mult: $a \cdot \langle x \rangle = \langle a \cdot x \rangle$
- Constant point mult: $[a] \cdot X = \langle a \cdot x \rangle$, where $X = x \cdot G$ (note that x may be unknown).

Threshold ECDSA signing in 3 phases

Key independent pre-processing

1. Use triples $([k], [b], [c])$ to compute $[k^{-1}]$
2. $\langle k \rangle = \mathsf{cnv}([k])$

Threshold ECDSA signing in 3 phases

Key independent pre-processing

1. Use triples $([k], [b], [c])$ to compute $[k^{-1}]$
2. $\langle k \rangle = \mathsf{cnv}([k])$

Message independent pre-processing

1. $[\mathsf{sk}'] = [\mathsf{sk} \cdot k^{-1}] = [\mathsf{sk}] \cdot [k^{-1}]$

Threshold ECDSA signing in 3 phases

Key independent pre-processing

1. Use triples $([k], [b], [c])$ to compute $[k^{-1}]$
2. $\langle k \rangle = \mathsf{cnv}([k])$

Message independent pre-processing

1. $[\mathsf{sk}'] = [\mathsf{sk} \cdot k^{-1}] = [\mathsf{sk}] \cdot [k^{-1}]$

Signing (input is $(\langle k \rangle, [\mathsf{sk}'], M)$)

1. $(r_x, r_y) = R = \mathsf{Open}(\langle k \rangle)$
2. $[s] = H(M) \cdot [k^{-1}] + r_x \cdot [\mathsf{sk}']$
3. $s = \mathsf{Open}([s])$, output (r_x, s)

Threshold ECDSA signing in 3 phases

Key independent pre-processing

1. Use triples $([k], [b], [c])$ to compute $[k^{-1}]$
2. $\langle k \rangle = \mathsf{cnv}([k])$

Message independent pre-processing

1. $[\mathsf{sk}'] = [\mathsf{sk} \cdot k^{-1}] = [\mathsf{sk}] \cdot [k^{-1}]$

Signing (input is $(\langle k \rangle, [\mathsf{sk}'], M)$)

1. $(r_x, r_y) = R = \mathsf{Open}(\langle k \rangle)$
2. $[s] = H(M) \cdot [k^{-1}] + r_x \cdot [\mathsf{sk}']$
3. $s = \mathsf{Open}([s])$, output (r_x, s)

Threshold ECDSA signing in 3 phases

Key independent pre-processing

1. Use triples $([k], [b], [c])$ to compute $[k^{-1}]$
2. $\langle k \rangle = \mathsf{cnv}([k])$

Message independent pre-processing

1. $[\mathsf{sk}'] = [\mathsf{sk} \cdot k^{-1}] = [\mathsf{sk}] \cdot [k^{-1}]$

Signing (input is $(\langle k \rangle, [\mathsf{sk}'], M)$)

1. $(r_x, r_y) = R = \mathsf{Open}(\langle k \rangle)$
2. $[s] = H(M) \cdot [k^{-1}] + r_x \cdot [\mathsf{sk}']$
3. $s = \mathsf{Open}([s])$, output (r_x, s)

Key generation just generate random $[x]$ and $\mathsf{pk} = \mathit{Open}(\mathsf{cnv}([x]))$

Benchmarks

Comparison with prior work

		LAN		WAN	
	n	Sign(ms)	KeyGen(ms)	Sign(ms)	KeyGen(ms)
Rep3	3	2.78	1.45	367.87	291.32
Shamir	3	3.02	1.39	1140.09	486.82
Mal. Rep3	3	3.45	1.57	1128.01	429.47
Mal. Shamir	3	4.43	1.89	2340.53	485.11
MASCOT	2	6.56	4.32	2688.92	2632.07
MASCOT–	2	3.61	4.41	729.08	2654.59
DKLS	2	3.58	43.73	234.37	1002.97
Unbound	2	11.33	315.96	490.73	1010.98
Kzen †	2	310.71	153.87	14441.83	7237.93

†: Implementation of [GG18] Fast Multiparty Threshold ECDSA with Fast Trustless Setup (CCS '18)

Benchmarks

Throughput

	LAN		WAN	
	Tuples per sec.	Sign (ms)	Tuples per sec.	Sign (ms)
Rep3	922.27	2.49	715.54	247.13
Shamir	1829.69	2.37	402.88	271.80
Mal. Rep3	914.65	2.52	309.76	245.14
Mal. Shamir	1792.30	2.91	172.87	416.60
MASCOT	380.19	4.82	31.98	756.34
MASCOT–	700.94	2.75	68.31	258.85