Efficient Online-friendly Two-Party ECDSA

Haiyang Xue

Joint work with Man Ho Au, Xiang Xie, Tsz Hon Yuen, and Handong Cui

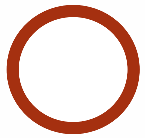
To appear in ACM CCS 2021

Outline

- Two-Party ECDSA
- Our Contribution
  - Generic Two-Party ECDSA from a single MtA
- Technical overview
- Instantiations and implementation
  - Paillier
  - CL encryption
  - OT

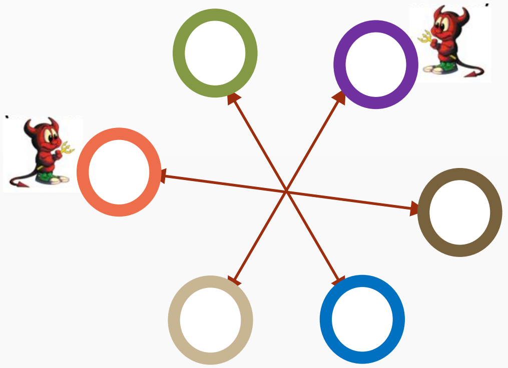
Motivation of distributed ECDSA

- ECDSA
  - Digital Signature Standard using Elliptic Curve Cryptography
  - Widely deployed, such as Bitcoin etc.
  - Stealing signing key means financial loss etc. (single-point of failure)



How to address single-points of failure？

- Distributed （Threshold） ECDSA
  - Protect the key by sharing among multiple parties
  - Such that no fewer user ($< t$) could generate a valid ECDSA



The threshold approach

Motivation of distributed ECDSA

- Threshold Cryptography Project at NIST
  - Scope: standardization of threshold schemes

NIST

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

PROJECTS

Multi-Party Threshold Cryptography MPTC

Overview

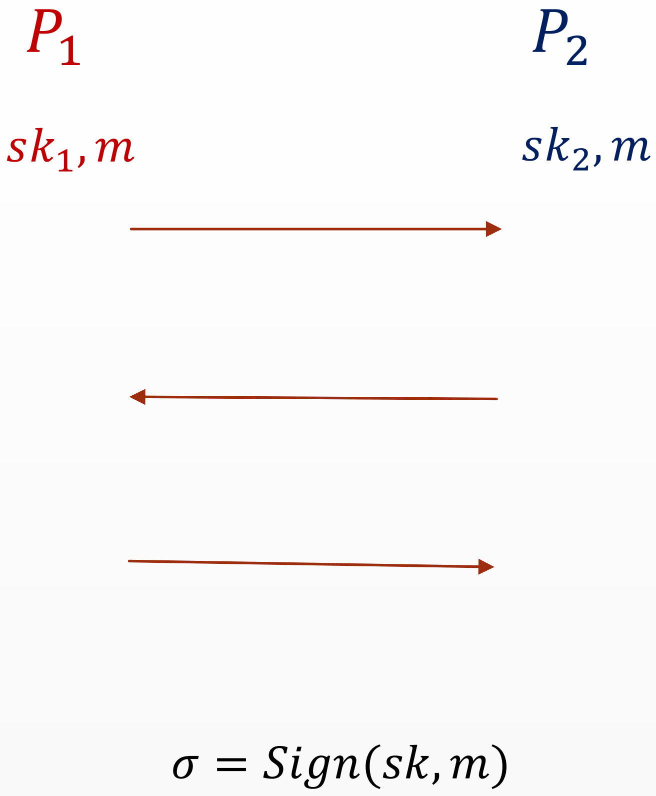
The multiparty paradigm of threshold cryptography enables a secure distribution of trust in the operation of cryptographic primitives. This can apply, for example, to the operations of key generation, signing, encryption and decryption.

4.1.2.2 ECDSA signature. A technical difficulty in threshold ECDSA is jointly computing a secret sharing of a multiplicative inverse of an additively-shared secret value. This is less straight-

https://csrc.nist.gov/projects/threshold-cryptography ; NIST.IR.8214A

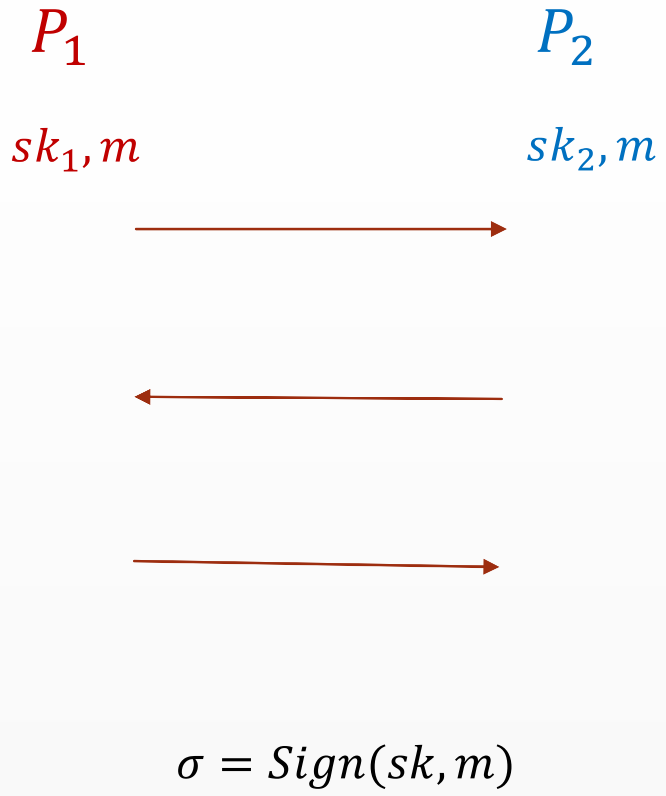
Two-Party Signature (with $t = 2$)

- **Setup:** The signing key is secret shared across 2 parties
- **Interaction:** The parties may collaborate, but their key shares remain secret
- **Correctness:** sign a message in a threshold manner
- **Security:**
  - Any P_i can not forge signature alone, or learn anything on sk
  - Reduce to the security of original signature



Two-Party Signature (with $t = 2$)

- **Setup:** The signing key is secret shared across 2 parties
- **Interaction:** The parties may collaborate, but their key shares remain secret
- **Correctness:** sign a message in a threshold manner
- **Security:**
  - Any P_i can not forge signature alone, or learn anything on sk
  - Reduce to the security of original signature



- Efficient Two-Party Schnorr since 90s

- Two-Party ECDSA is much more challenging

Challenge in Two-Party ECDSA: ECDSA

Public parameters: $G =< P >$ with prime order q

Secret signing key: $x \leftarrow Z_q$

Public key: $Q = x \cdot P$

ECDSA Algorithm

- **Sign**(x, m)
  - $R = k \cdot P$ where $k \leftarrow Z_q$; $r = r_x$ where $R = (r_x, r_y)$
  - $s = k^{-1}(H(m) + x \cdot r) \bmod q$
  - Output (r, s)
- **Verify**(r, s)
  - $(r_x, r_y) = s^{-1}[H(m) \cdot P + r \cdot Q]$
  - $r =? r_x$

Challenge in Two-Party ECDSA: Schnorr

Public parameters: $G =< P >$ with prime order q

Secret signing key: $x \leftarrow Z_q$

Public key: $Q = x \cdot P$

Schnorr Algorithm

- **Sign**(x, m)
  - $R = k \cdot P$ where $k \leftarrow Z_q$; $r = r_x$ where $R = (r_x, r_y)$
  - $s = k + x \cdot H(R|m) \bmod q$
  - Output (r, s)
- **Verify**(r, s)
  - $s \cdot P =? R + H(R|m) \cdot P$

Challenge in Two-Party ECDSA

Public parameters: $G =< P >$ with prime order q

Secret signing key: $x \leftarrow Z_q$

Public key: $Q = x \cdot P$

Schnorr Algorithm

- $R = k \cdot P$ where $k \leftarrow Z_q$
- $r = r_x$ where $R = (r_x, r_y)$
- $s = k + x \cdot H(R|m) \bmod q$
- Output (r, s)

ECDSA Algorithm

- $R = k \cdot P$ where $k \leftarrow Z_q$
- $r = r_x$ where $R = (r_x, r_y)$
- $s = k^{-1}(H(m) + x \cdot r) \bmod q$
- Output (r, s)

Challenge in Two-Party ECDSA

Public parameters: $G =< P >$ with prime order q

Secret signing key: $x \leftarrow Z_q$

Public key: $Q = x \cdot P$

Schnorr Algorithm

- $R = (k_1 + k_2) \cdot P$
- $r = r_x$
- $s = k_1 + x_1 \cdot H(R|m) + k_2 + x_2 \cdot H(R|m)$
- Output (r, s)

ECDSA Algorithm

- $R = k \cdot P$ where $k \leftarrow Z_q$
- $r = r_x$ where $R = (r_x, r_y)$
- $s = k^{-1}(H(m) + x \cdot r) \bmod q$
- Output (r, s)

Using additive share of x and k

$x = x_1 + x_2$

$k = k_1 + k_2$

Compute k^{-1} and $k^{-1}x$ from shares of x and k

Challenge in Two-Party ECDSA

Public parameters: $G =< P >$ with prime order q

Secret signing key: $x \leftarrow Z_q$

Public key: $Q = x \cdot P$

ECDSA Algorithm

- $R = k \cdot P$ where $k \leftarrow Z_q$
- $r = r_x$ where $R = (r_x, r_y)$
- $s = k^{-1}(H(m) + x \cdot r) \bmod q$
- Output (r, s)

Compute k^{-1} and $k^{-1}x$ from shares of x and k

Malicious security for any circuit

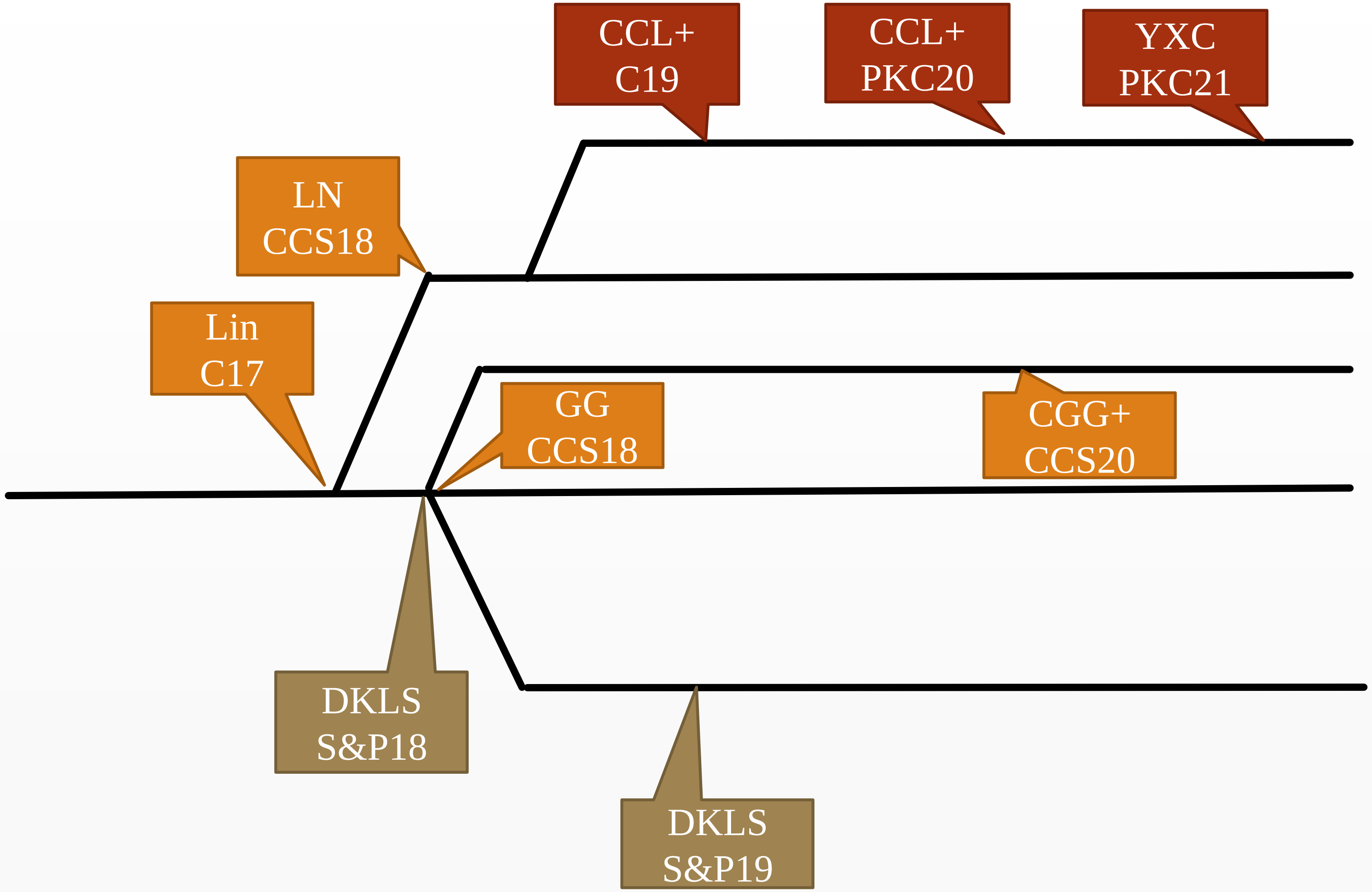
GMW compiler
Commitment
Zero-knowledge proof

Inefficient

Semi-honest security for any circuit
Goldreich-Micali-Wigderson (GMW)
Ben-Goldwasser-Wigderson(BGW)
etc.

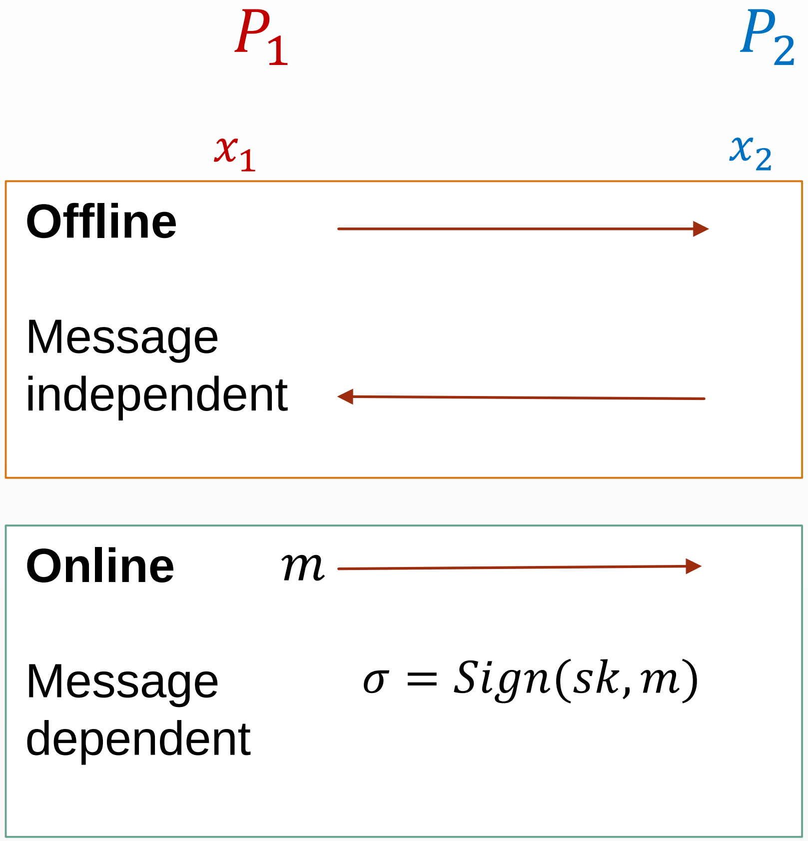
Previous works on Two-Party ECDSA

- Homomorphic Enc: CL
- Homomorphic Enc: Paillier
- Oblivious Transfer (OT)



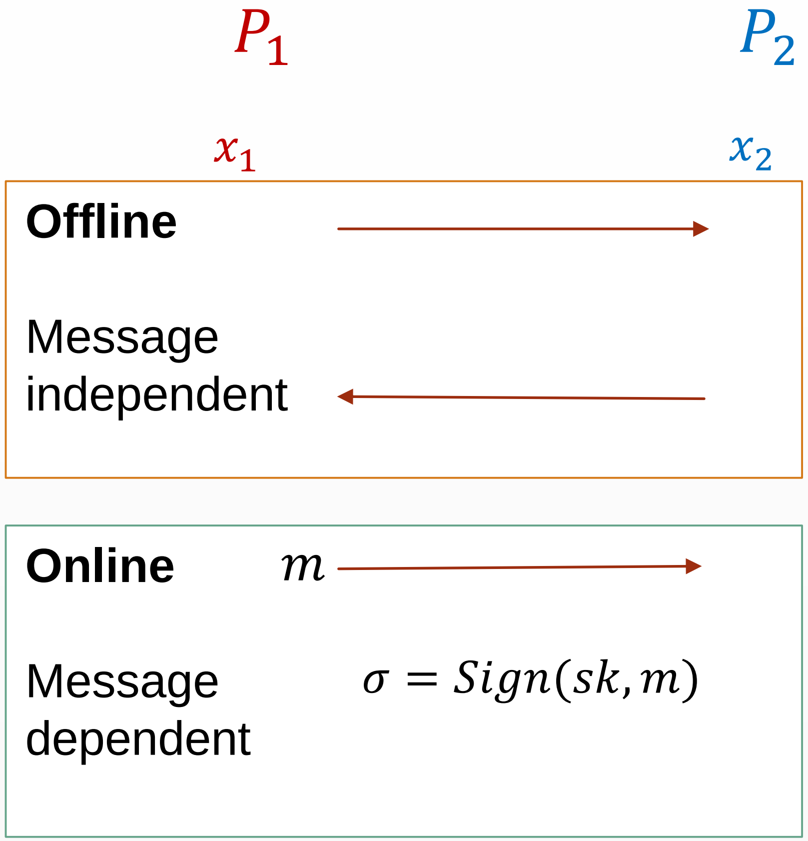
Previous works on Two-Party ECDSA

- The offline phase (aka. pre-processing) is message independent.
- We say the online phase of a two-party ECDSA is optimal if it is **non-interactive** and its cost is approximately **a verification procedure**.
- Two-party ECDSA is online-friendly if its online phase is **optimal**.



Previous works on Two-Party ECDSA

Schemes	Offline	Online
[Lin17, CCL+19]	Enc	Dec
[LN18]	2*MtA	MtA
[GG18, CCL+20,YXC21]	4*MtA	Fast
[DKLS18]	2~3*MtA	Optimal
[CGG+20, DKLS19]	4*MtA	Optimal



Previous works on Two-Party ECDSA

Schemes	Offline	Online
[Lin17, CCL+19]	Enc	Dec
[LN18]	2*MtA	MtA
[GG18, CCL+20,YXC21]	4*MtA	Fast
[DKLS18]	2~3*MtA	Optimal
[CGG+20, DKLS19]	4*MtA	Optimal

Paillier	~10ms	~3KB
CL	~200ms	~200B

Costly

Multi-to-Add protocol

Paillier	~200ms	~6KB
CL	~1300ms	~1KB
OT	cheap	~90KB

Motivation: online-friendly scheme with one MtA

Schemes	Offline	Online
[Lin17,CCL+19]	Enc	Dec
[LN18]	2*MtA	MtA
[GG18, CCL+20,YCX21]	4*MtA	Fast
[DKLS18]	2~3*MtA	Optimal
[CGG+20, DKLS19]	4*MtA	Optimal
Is it possible??	1*MtA	Optimal

Our contribution

Schemes	Offline	Online
[Lin17,CCL+19]	Enc	Dec
[LN18]	2*MtA	MtA
[GG18, CCL+20,YCX21]	4*MtA	Fast
[DKLS18]	2~3*MtA	Optimal
[CGG+20, DKLS19]	4*MtA	Optimal
This work: 2ECDSA	1*MtA	Optimal

Comparison

Signing Protocols	Computation		Communication		Passes
	offline	online	offline	online	
LNR18 [26]	28E + 157M (461ms)	14E + 121M (302ms)	$32\ell_N + 67\kappa$ (12KB)	$16\ell_N + 51\kappa$ (6.6KB)	8
GG18 [19]	42E + 40M (1237ms)	17M (3ms)	$40\ell_N + 18\kappa$ (15.5KB)	9κ (288B)	9
CGGMP20 [6]	208E + 44M (2037ms)	2M (0.2ms)	$118\ell_N + 20\kappa$ (44KB)	κ (32B)	4
2ECDSA (Paillier)	14E + 11M (226ms)	2M (0.2ms)	$16\ell_N + 11\kappa$ (6.3KB)	κ (32B)	3
Lin17 [25] (Paillier-EC)	2E + 8M (34ms)	1E + 2M (8ms)	12κ (192B)	$2\ell_N$ (768B)	3
GG18 [19] (Paillier-EC)	18E + 40M (360ms)	17M (3ms)	$16\ell_N + 18\kappa$ (6.6KB)	9κ (288B)	9
2ECDSA (Paillier-EC)	8E + 14M (141ms)	2M (0.2ms)	$10\ell_N + 12\kappa$ (4.1KB)	κ (32B)	3
CCLST19 [7]	4E + 8M (475ms)	1E + 2M (190ms)	6κ (208B)	14κ (505B)	3
CCLST20 [8]	28E + 8M (3316ms)	17M (3ms)	140κ (4.5KB)	9κ (288B)	8
YCX21 [33]	28E + 8M (4550ms)	17M (3ms)	140κ (4.5KB)	9κ (288B)	8
2ECDSA (CL)	11E + 11M (1386ms)	2M (0.2ms)	53κ (1.7KB)	κ (32B)	3
DKLS18 [15]	13M (2.9ms)	2M (0.2ms)	$16\kappa^2$ (169.8KB)	κ (32B)	2
DKLS19 [16]	13M (3.7ms)	2M (0.2ms)	$20\kappa^2$ (180KB)	κ (32B)	7
2ECDSA (OT)	11M (2.6ms)	2M (0.2ms)	$8\kappa^2$ (90.9KB)	κ (32B)	3

Technical Overview

Preliminary: Paillier and CL Encryption

- Additive Homomorphic Encryption Scheme:

$$\mathrm{Enc}(m_1 + m_2) = \mathrm{Enc}(m_1) \oplus \mathrm{Enc}(m_2)$$

$$\mathrm{Enc}(a \cdot m) = \mathrm{Enc}(m)^a = a \odot \mathrm{Enc}(m)$$

Schemes	over	Message Space
Paillier	Z_{N^2} (N is RSA modulus)	Z_N
CL Encryption	Class group	Z_q ($=\#G$)

Paillier Encryption

- Let $N = pq$ be RSA modulus.

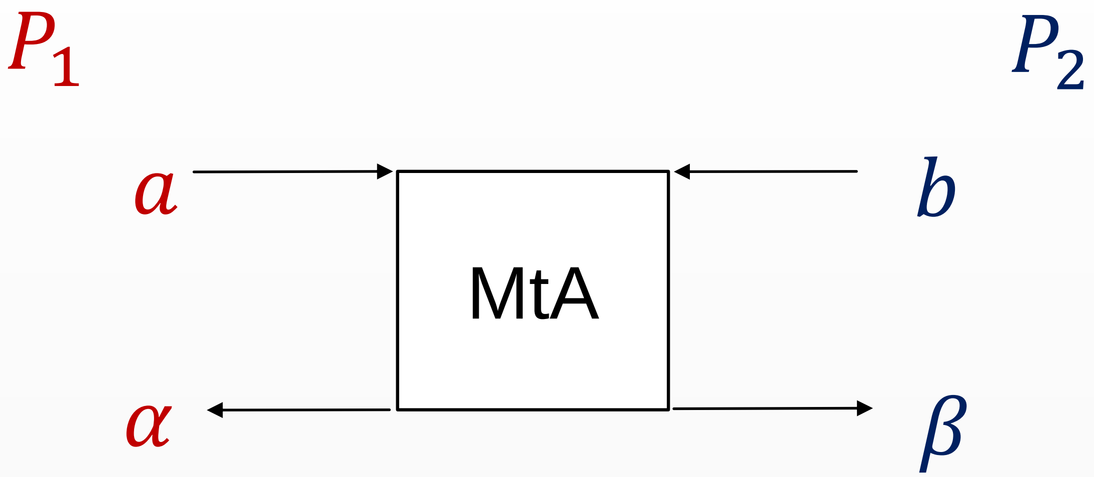
Secret key: p, q public key : N

$$\text{Enc}(N, m) = (1 + \text{N})^m \, r^N \, mod \, N^2$$

$$\text{Enc}(N, (m_1 + m_2) \, mod \, N) = \text{Enc}(N, m_1) \oplus \text{Enc}(N, m_2)$$

Preliminary: Multi-to-Add Protocol

- Multi-to-Add Protocol (MtA)



Such that $\alpha + \beta = a \cdot b \bmod q$

ECDSA

$$s = k^{-1}(H(m) + x \cdot r)$$

- $H(m)$ and r is publicly known to both parties
- x is the secret key
- k is the nonce

Lin17 and CCL+19

- Multiplicative share of $k = k_1 \cdot k_2$ and $x = x_1 \cdot x_2$
- Goal:

$$s = k_1^{-1} \cdot \underbrace{k_2^{-1} \left(H(m) + x_1 \cdot x_2 \cdot r\right)}_{s_1}$$

- If P_1 has sent $\mathrm{Enc}\,(x_1)$ to P_2 in the Key Generation phase
- On receiving message m, P_2 could compute

$$\mathrm{Enc}(s_1) = \mathrm{Enc}\left(k_2^{-1}(H(m) + x_1 \cdot x_2 \cdot r)\right)$$

- With decryption key, P_1 could compute s_1 and then s.

Lin17 and CCL+19

- Multiplicative share of $k = k_1 \cdot k_2$ and $x = x_1 \cdot x_2$
- Goal:

$$s = k_1^{-1} \cdot \underbrace{k_2^{-1}\,(H(m) + x_1 \cdot x_2 \cdot r)}_{s_1}$$

- If P_1 has sent $\mathrm{Enc}\,(x_1)$ to P_2 in the Key Generation phase

However, decryption is required in the online phase;

- Furthermore, non-standard assumption is required, such as Paillier-EC

On receiving message ..., P_2 could compute

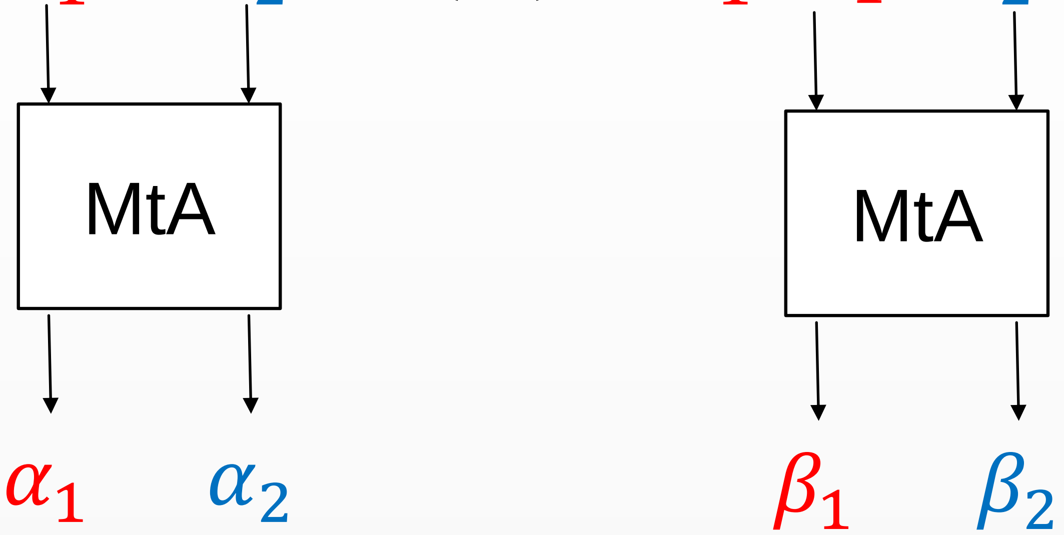
$$\mathrm{Enc}(s_1) = \mathrm{Enc}\,(k_2^{-1}(H(m) + x_1 \cdot x_2 \cdot r))$$

- With decryption key, P_1 could compute s_1 and then s.

DKLS18

- Multiplicative share of $k = k_1 \cdot k_2$ and $x = x_1 \cdot x_2$
- Goal:

$$s = k_1^{-1} \cdot k_2^{-1} \, H(m) + k_1^{-1} x_1 \cdot k_2^{-1} x_2 \cdot r$$



$$s = (\alpha_1 + \alpha_2) H(m) + (\beta_1 + \beta_2) \cdot r$$

- Two MtA are required.

LNR18 etc…

- Additive share of $k = k_1 + k_2$ and $x = x_1 + x_2$
- Goal:

$$s = (k_1 + k_2)^{-1}[H(m) + (x_1 + x_2) \cdot r]$$

Step 1

2*MtA

α_1 α_2

$$s = (\alpha_1 + \alpha_2)H(m) + (\alpha_1 + \alpha_2)(x_1 + x_2) \cdot r$$

Step 2

2*MtA

$$s = (\alpha_1 + \alpha_2)H(m) + (\beta_1 + \beta_2) \cdot r$$

LNR18 etc…

- Additive share of $k = k_1 + k_2$ and $x = x_1 + x_2$
- Goal:

$$s = (k_1 + k_2)^{-1}[H(m) + (x_1 + x_2) \cdot r]$$

2*MtA

α_1 α_2

$$s = (\alpha_1 + \alpha_2)H(m) + (\alpha_1 + \alpha_2)(x_1 + x_2) \cdot r$$

2*MtA

- 4 MtA are required.

$$s = (\alpha_1 + \alpha_2)H(m) + (\beta_1 + \beta_2) \cdot r$$

Our Construction with one MtA

- We start from share of $k = k_1 \cdot k_2$ and $x = x_1 + x_2$
- Goal:

$$s = k_1^{-1} \cdot k_2^{-1}[H(m) + (x_1 + x_2) \cdot r]$$

If P_1, P_2 can corporately compute x_1', x_2' such that

$$x_1 + x_2 = x_1' k_2 + x_2'$$

then

$$s = k_1^{-1} \cdot \underbrace{[k_2^{-1}(H(m) + r x_2')}_{P_2 \text{ could compute by itself}} + r x_1']$$

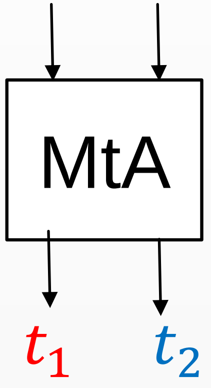
P_2 could compute by itself

Our Construction with one MtA

- We start from share of $k = k_1 \cdot k_2$ and $x = x_1 + x_2$

All we need: P_1, P_2 compute x_1', x_2' such that

$$x_1 + x_2 = x_1' k_2 + x_2'$$



Then, $x_2' = x_2 - t_2 + (x_1 + t_1)$

One-time Padding

- Only one MtA is required

Instantiations

MtA from Paillier

	P_1		P_2
Setup			
		$\xleftarrow{N, \pi_P}$	$pk = N, sk = \phi(N)$
Multiplication			
	$\alpha' \leftarrow \mathbb{Z}_K$	$\xleftarrow{c_B, \pi_B}$	$c_B = \mathsf{Enc}(pk, b)$
	$c_A = a \odot c_B \oplus \mathsf{Enc}(pk, \alpha')$	$\xrightarrow{c_A, \pi_A}$	
	$\alpha = -\alpha' \mod q$		$\beta = \mathsf{Dec}(sk, c_A) \mod q$

- Enc is the Paillier encryption
- π_P, π_B, π_A is the zero-knowledge proof for the correctness generation of N, c_B, c_A respectively

Paillier-based Two-Party ECDSA

Schemes	Computation		Communication	
	Offline	Online	Offline	Online
LNR18 [25]	461ms	302ms	12.1KB	6.6KB
GG18 [18]	1237ms	3ms	15.5KB	288B
CGGMP20 [6]	2037ms	0.2ms	44KB	32B
2ECDSA(Paillier)	226ms	0.2ms	6.3KB	32B
Lin17 [24]	34ms	8ms	192B	768B
GG18(Paillier-EC)[18]	360ms	3ms	6.6KB	288B
2ECDSA(Paillier-EC)	141ms	0.2ms	4.1KB	32B

Table 3: Cost comparison of Paillier-based schemes.

MtA from CL encryption

$P_1(\mathsf{pk}, a)$		$P_2(\mathsf{pk},\ \mathsf{sk}; b)$
$\alpha' \leftarrow \mathbb{Z}_q$	$\xleftarrow{c_B, \pi_{CL}}$	$c_B = \mathsf{Enc}_{cl}(\mathsf{pk}, b)$
$c_A = a \odot c_B \oplus \mathsf{Enc}_{cl}(pk, \alpha')$	$\xrightarrow{c_A}$	
$\alpha = -\alpha' \mod q$		$\beta' = \mathsf{Dec}_{cl}(\mathsf{sk}, c_A)$
		$\beta = \beta' \mod q$

- Enc_{cl} is the CL encryption over class group
- π_{CL} is the zero-knowledge proof for the correctness generation of c_B respectively

CL-based Two-Party ECDSA

Schemes	Computation		Communication	
	Offline	Online	Offline	Online
CCLST19 [7]	475ms	190ms	505B	208B
CCLST20 [8]	3316ms	3ms	4.5KB	288B
YCX21 [31]	4550ms	3ms	4.5KB	288B
2ECDSA(CL)	1386ms	0.2ms	1.7KB	32B

Table 5: Cost comparison of CL-based schemes.

MtA from Oblivious Transfer (OT)

OT is a fundamental primitive of multiparty computation (MPC).

MtA from Oblivious Transfer (OT)

Sender		**Receiver**
Input: (M_0, M_1)		Input: c
Output: none		Output: M_c
$a \leftarrow \mathbb{Z}_p$		$b \leftarrow \mathbb{Z}_p$
	$A = g^a$ $\longrightarrow$	
		if $c = 0$: $B = g^b$
		if $c = 1$: $B = Ag^b$
	$\longleftarrow$ B	
$k_0 = H(B^a)$		$k_R = H(A^b)$
$k_1 = H\left(\left(\frac{B}{A}\right)^a\right)$		
	$e_0 \leftarrow E_{k_0}(M_0)$	
	$e_1 \leftarrow E_{k_1}(M_1)$	
	$\longrightarrow$	
		$M_c = D_{k_R}(e_c)$

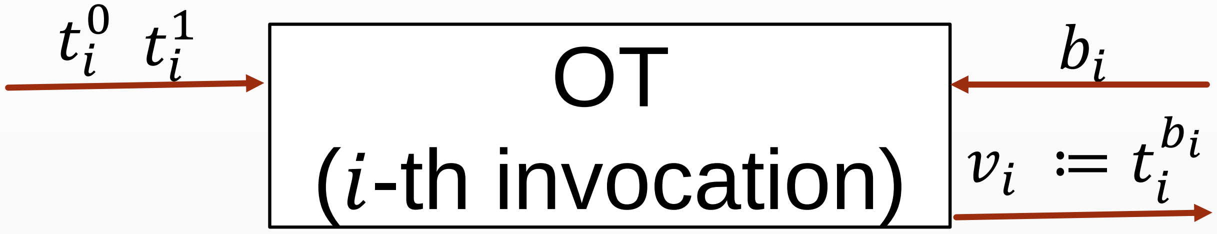
MtA from Oblivious Transfer (OT)

P_2 (a)

Randomly pick $s_0, \dots, s_{n-1}$

For each i, define $t_i^0 = 2^i a + s_i$; $t_i^1 = s_i$

P_1 $(b := b_0, b_1, \dots, b_{n-1})$



$\alpha = -\sum s_i$

$\beta = \sum v_i$

Note: $\alpha + \beta = ab$

OT

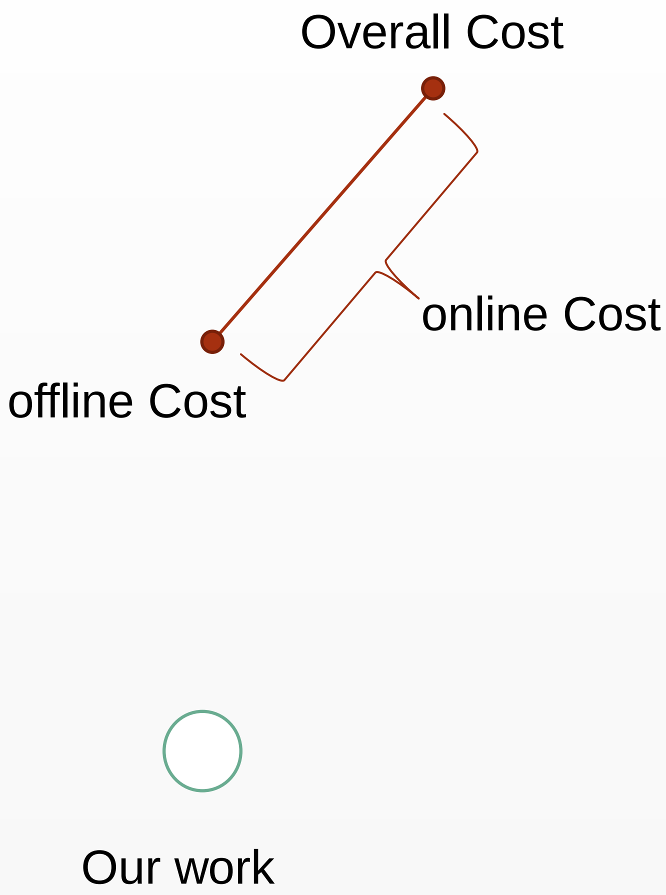
Schemes	Computation		Communication	
	Offline	Online	Offline	Online
DKLS18 [15]	2.9ms	0.2ms	169.8KB	32B
DKLS19 [16]	3.7ms	0.2ms	180KB	32B
2ECDSA(OT)	2.6ms	0.2ms	90.9KB	32B

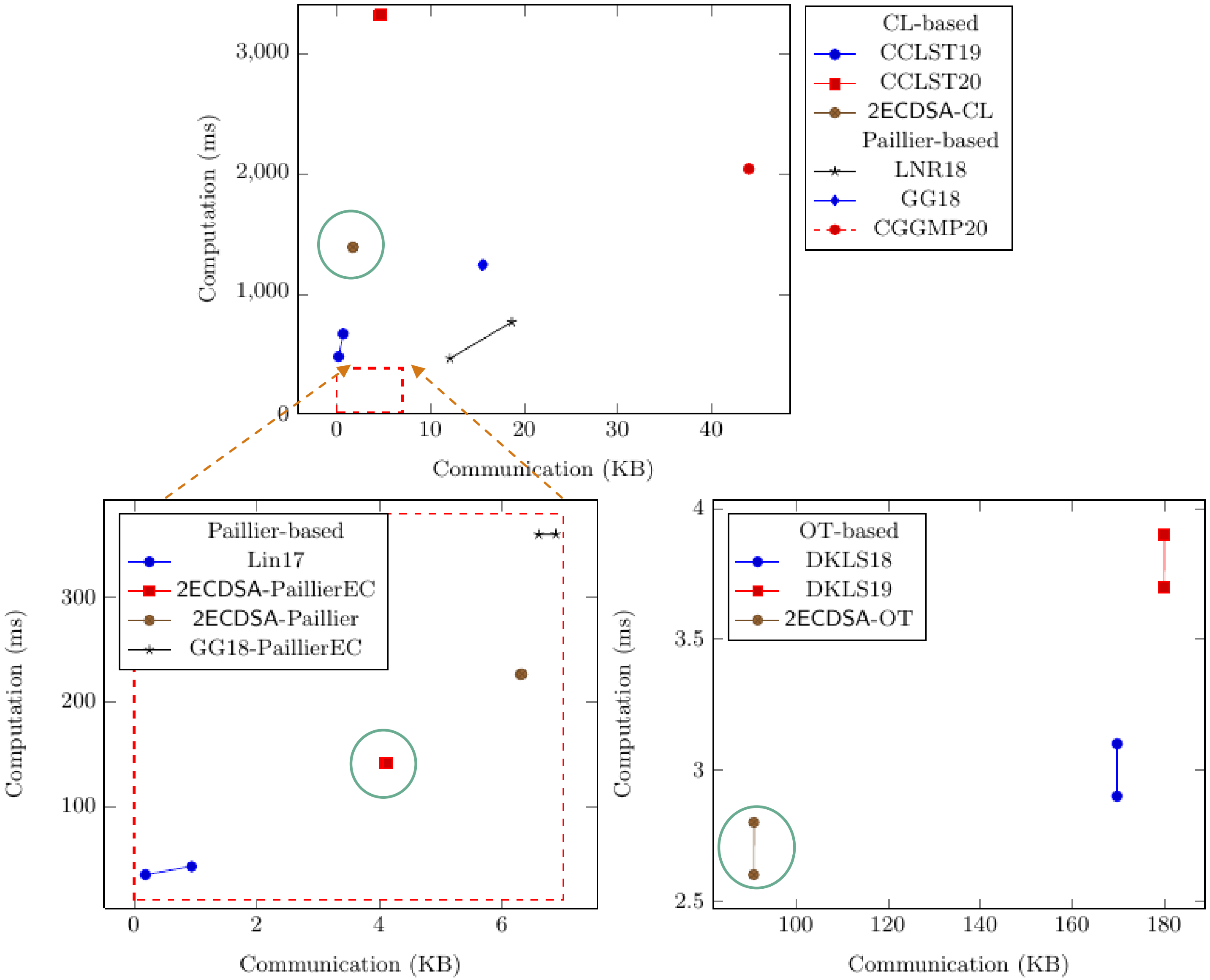
Table 4: Cost comparison of OT-based schemes.

MtAs from HE vs OT

OT-based MtA	Paillier/CL-based MtA
High communication	Low communication
Low computation	High computation
No zero-knowledge proof	zero-knowledge proof
No extra assumption	May need extra assumptions

Comparison in one figure



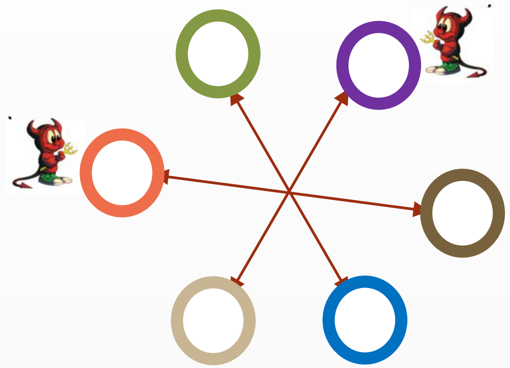


Conclusion

- We propose a online-friendly two-party ECDSA such that
  - its online computation is extremely fast
  - and its offline phase just need a single execution of MtA
- Our scheme could be instantiated with Paillier/CL encryption and OT

Following works: t-out-of-n ECDSA

- This work only supports two-party, i.e., 2-out-of-n.
- How about t-out-of-n ECDSA?



The threshold approach

One more thing: SM2

Public parameters: $G =< P >$ with prime order q

Secret signing key: $x \leftarrow Z_q$

Public key: $Q = x \cdot P$

SM2 Algorithm

- **Sign**(x, m)
  - $R = k \cdot P$ where $k \leftarrow Z_q$; $r = r_x$ where $R = (r_x, r_y)$
  - $s = x^{-1}[k + r + H(m)] \bmod q$
  - Output (r, s)
- **Verify**(r, s)
  - $s \cdot Q =? R + r \cdot P + H(m) \cdot P$

Thanks

Q & A

Reference

- [CCL+19] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2019. Two-party ECDSA from hash proof systems and efficient instantiations. CRYPTO 2019
- [CCL+20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2020. Bandwidth-efficient threshold ECDSA. PKC 2020
- [CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts. ACM CCS 2020
- [DKLS18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. IEEE S&P 2018
- [DKLS19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat.Threshold ECDSA from ECDSA assumptions: the multiparty case. IEEE S&P 2019
- [GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. ACM CCS 2018
- [Lin17] Yehuda Lindell. Fast secure two-party ECDSA signing. CRYPTO 2017
- [LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. ACM CCS 2018
- [YCX21] Tsz Hon Yuen, Handong Cui, and Xiang Xie. Compact Zero-Knowledge Proofs for Threshold ECDSA with Trustless Setup. PKC 2021