Secure Two-party Threshold ECDSA from ECDSA Assumptions - 2018

Doerner, Kondi, Lee, shelat (DKLs)

Selman Pınar
TÜBİTAK - BİLGEM

November 21, 2025

Overview

- ECDSA
- Secret Sharing
- Oblivious Transfer
- The Simplest OT
- Beaver MtA with OT
- VSOT
- Corallated OT
- KOS protokolü
- MtA
- İmzalama Şemaları

Algorithm 1. $\mathsf{Gen}(1^\kappa)$**:**

1) Uniformly choose a secret key $\mathsf{sk} \leftarrow \mathbb{Z}_q$.
2) Calculate the public key as $\mathsf{pk} := \mathsf{sk} \cdot G$.
3) Output $(\mathsf{pk}, \mathsf{sk})$.

Algorithm 2. $\mathsf{Sign}(\mathsf{sk} \in \mathbb{Z}_q, m \in \{0,1\}^*)$**:**

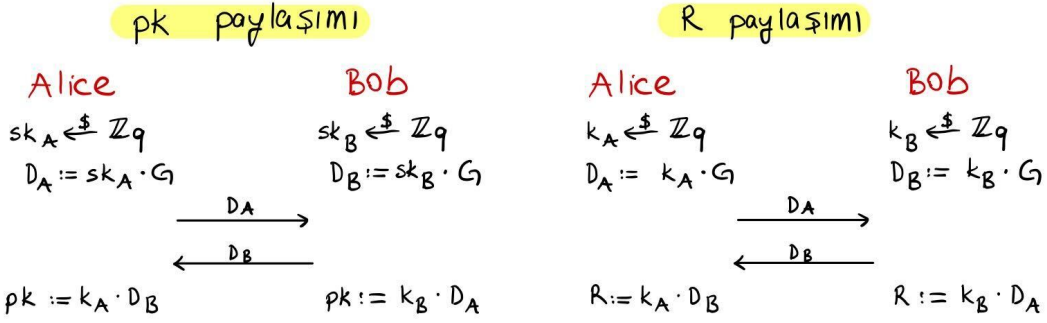
1) Uniformly choose an instance key $k \leftarrow \mathbb{Z}_q$.
2) Calculate $(r_x, r_y) = R := k \cdot G$.
3) Calculate
$$\mathsf{sig} := \frac{H(m) + \mathsf{sk} \cdot r_x}{k}$$
4) Output $\sigma := (\mathsf{sig} \mod q, r_x \mod q)$.

Algorithm 3. $\mathsf{Verify}(\mathsf{pk} \in \mathbb{G}, m, \sigma \in (\mathbb{Z}_q, \mathbb{Z}_q))$**:**

1) Parse σ as (sig, r_x).
2) Calculate
$$(r'_x, r'_y) = R' := \frac{H(m) \cdot G + r_x \cdot \mathsf{pk}}{\mathsf{sig}}$$
3) Output 1 if and only if $(r'_x \mod q) = (r_x \mod q)$.

Public Key ve R paylaşımı

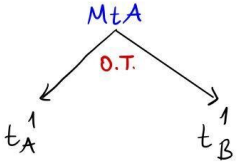
- pk ve R değerleri DH exchange ile oluşturulur.

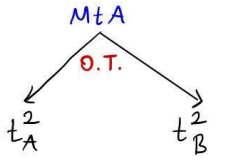


(2,2) DKLs Mantığı

$k = k_A \cdot k_B$ $\qquad$ $sk = sk_A \cdot sk_B$

$$sig = H(m) \cdot \left(\frac{1}{k_A} \cdot \frac{1}{k_B} \right) + r_x \left(\frac{sk_A}{k_A} \cdot \frac{sk_B}{k_B} \right)$$





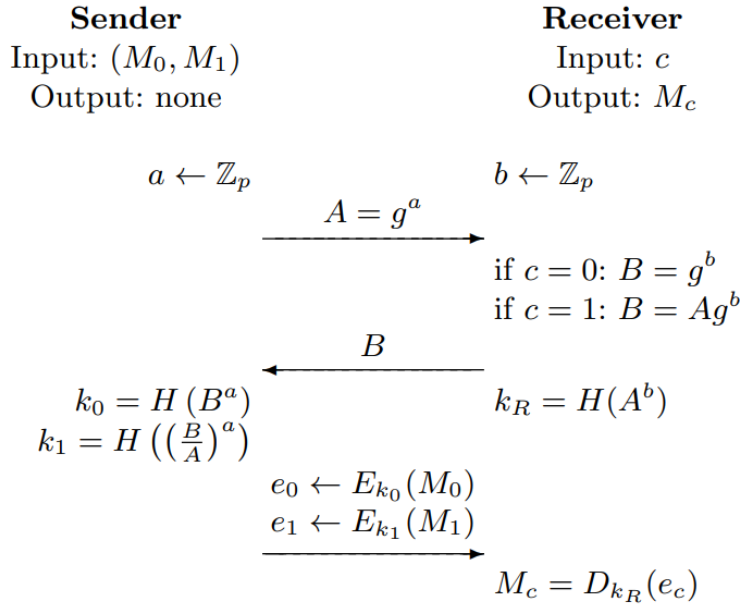
$t_A^1 + t_B^1 = \frac{1}{k_A} \cdot \frac{1}{k_B}$ $\qquad$ $t_A^2 + t_B^2 = \frac{sk_A}{k_A} \cdot \frac{sk_B}{k_B}$

$sig_A = H(m) \cdot t_A^1 + r_x \cdot t_A^2$ $\qquad$ $sig_B = H(m) \cdot t_B^1 + r_x \cdot t_B^2$

$sig = sig_A + sig_B$

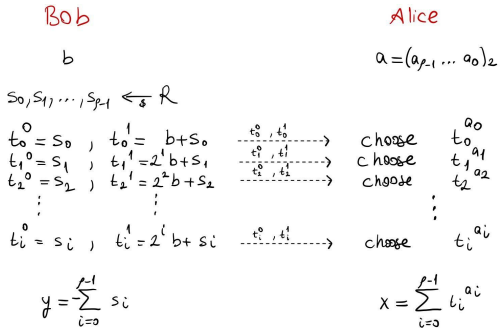
OT functionality

Sender		**Receiver**
Input: (m_0, m_1)		Input: $b \in \{0, 1\}$
	$(m_0, m_1) \longrightarrow$	
Output: nothing, $\perp$		Output: m_b
		Does not know: m_{1-b}


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

Beaver MtA Tecnique with OT

Protokol:

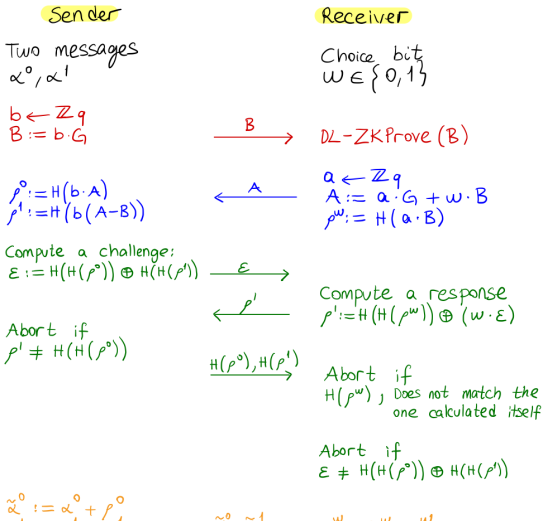


İspat:

$$
\begin{aligned}
x + y &= \sum_{i=0}^{\rho-1} t_i^{a_i} - \sum_{i=0}^{\rho-1} s_i \\
&\equiv \sum_{i=0}^{\rho-1} (a_i \cdot 2^i b + s_i) - \sum_{i=0}^{\rho-1} s_i \\
&\equiv b \sum_{i=0}^{\rho-1} a_i \cdot 2^i \\
&\equiv ab
\end{aligned}
$$

Halledilmesi gereken sıkıntılar

- DH anahtar değişiminde anahtar parçasını diğer kişiden alan ilk kişi, ortak anahtarı manipüle edebilir.
  Çözüm → Modified DH ve Commitment Scheme
- DH anahtar değişiminde partilerden biri karşı tarafa, elinde olan gizli anahtara denk gelen farklı açık anahtar gönderebilir.
  Çözüm → Discreate Logarithm Zero Knowladge Proof
- OT sırasında gönderici bilerek yanlış mesaj gönderip protokolün doğru çalışıp çalışmadığına göre karşı tarafın bitini öğrenebilir. (Selective failure attack)
  Çözüm → Encoding Scheme
- MtA protokolünde partilerden biri girdileri yanlış girebilir.
  Çözüm → Maskeleme


Sender
Receiver
Two messages
α^0, α^1
Choice bit
$w \in \{0,1\}$
$b \leftarrow \mathbb{Z}_q$
$B := b \cdot G$
B
DL-ZKProve(B)
$\rho^0 := H(b \cdot A)$
$\rho^1 := H(b(A-B))$
A
$a \leftarrow \mathbb{Z}_q$
$A := a \cdot G + w \cdot B$
$\rho^w := H(a \cdot B)$
Compute a challenge:
$\varepsilon := H(H(\rho^0)) \oplus H(H(\rho^1))$
ε
Compute a response
$\rho' := H(H(\rho^w)) \oplus (w \cdot \varepsilon)$
ρ'
Abort if
$\rho' \neq H(H(\rho^0))$
$H(\rho^0), H(\rho^1)$
Abort if
$H(\rho^w)$, Does not match the one calculated itself
Abort if
$\varepsilon \neq H(H(\rho^0)) \oplus H(H(\rho^1))$
$\tilde{\alpha}^0 := \alpha^0 + \rho^0$
$\tilde{\alpha}^1 := \alpha^1 + \rho^1$
$\tilde{\alpha}^0, \tilde{\alpha}^1$
$\alpha^w = \tilde{\alpha}^w - \rho^w$
Public Key
Pad Transfer
Verification
Message Transfer

Corallated OT - $\mathcal{F}_{\text{COTe}}^{\ell}$ fonksiyonu

Sender

$\alpha \in G_1 \times G_2 \times \cdots \times G_\ell$: correlation

$t_S \xleftarrow{\$} G_1 \times G_2 \times \cdots \times G_\ell$

Output t_S

Receiver

$w \in \{0,1\}^\ell$: choice

if $w_i = 0 \longrightarrow t_{S_i}$

if $w_i = 1 \longrightarrow \alpha_i - t_{S_i}$

which means :

$t_R : \{ w_i \cdot \alpha_i - t_{S_i} \}_{i \in [1,\ell]}$

Output t_R

KOS Setup

- KOS Setup ve KOS Extension protokolleri yukarıdaki $\mathcal{F}_{\text{COTe}}^{\ell}$ fonksiyonunu gerçekleştirir.

KOS SETUP PROTOKOLÜ

Alice - Receiver	**Bob - Sender**
Corellation vector: $\Delta \leftarrow \{0,1\}^{\kappa}$	Two random seed elements: $s_i^0 \leftarrow \mathbb{Z}_q$ $s_i^1 \leftarrow \mathbb{Z}_q$
For all i take: $s_i^{\Delta_i}$	
Output: $s^{\Delta} = \{s_i^{\Delta_i}\}_{i \in [1,\kappa]}$	Output: $s^0 = \{s_i^0\}_{i \in [1,\kappa]}$ $s^1 = \{s_i^1\}_{i \in [1,\kappa]}$

KOS Extension

Alice inputs: $\alpha \in G_1 \times G_2 \times \cdots \times G_\ell$, Δ and s^Δ	**Bob** inputs: $\boldsymbol{\omega} \in \{0,1\}^\ell$, s^0 and s^1
	$\boldsymbol{\gamma}^{\text{ext}} \leftarrow \{0,1\}^{\kappa^{\text{OT}}}$ $\mathbf{w} := \boldsymbol{\omega} \| \boldsymbol{\gamma}^{\text{ext}}$ $w := \text{Bits}^{-1}(\mathbf{w}) \in \mathbb{Z}_{2^{\ell'}}$
$\mathbf{v}^\nabla := \left(\text{PRG}(\mathbf{s}_i^\nabla)\right)_{i\in[1,\kappa]}$	$\mathbf{v}^0 := \left(\text{PRG}(\mathbf{s}_i^0)\right)_{i\in[1,\kappa]}$ $\mathbf{v}^1 := \left(\text{PRG}(\mathbf{s}_i^1)\right)_{i\in[1,\kappa]}$
	$\boldsymbol{\psi} := \text{transpose}(\mathbf{v}^0)$ $\mathbf{u} := \{\mathbf{v}_i^0 \oplus \mathbf{v}_i^1 \oplus w\}_{i\in[1,\kappa]}$ $\boldsymbol{\chi} := \{H(j\|\mathbf{u})\}_{j\in[1,\ell']}$ $w' := \bigoplus_{j\in[1,\ell']} \mathbf{w}_j \cdot \boldsymbol{\chi}_j$ $\psi' := \bigoplus_{j\in[1,\ell']} \boldsymbol{\psi}_j \wedge \boldsymbol{\chi}_j$ Send these trible to Alice $(w', \psi', \mathbf{u})$

$\mathbf{z} := \{\mathbf{v}_i^{\Delta_i} \oplus (\Delta_i \cdot \mathbf{u}_i)\}_{i\in[1,\kappa]}$ $\zeta := \mathsf{transpose}(\mathbf{z})$ $\chi := \{H(j\|\|\mathbf{u})\}_{j\in[1,\ell']}$ $z' := \bigoplus_{j\in[1,\ell']} \zeta_j \wedge \chi_j$ If $z' \neq \psi' \oplus (\Delta \wedge w')$ abort	
$\mathbf{t}_A := \{H^{\|\alpha_j\|/\kappa}(j\|\|\zeta_j)\}_{j\in[1,\ell]}$ $\tau := \{H^{\|\alpha_j\|/\kappa}(j\|\|(\zeta_j \oplus \nabla)) - \mathbf{t}_{Aj} + \alpha_j\}_{j\in[1,\ell]}$ Send τ to Bob	
	$\mathbf{t}_B := \begin{cases} -H^{\|\tau_j\|/\kappa}(j\|\|\psi_j) & \text{eğer } \mathbf{w}_j = 0 \\ \tau_j - H^{\|\tau_j\|/\kappa}(j\|\|\psi_j) & \text{eğer } \mathbf{w}_j = 1 \end{cases}_{j\in[1,\ell]}$

π_{Mul} Protokolü

- Bu protokol Alice ve Bob'dab sırasıyla α ve β değerlerini alıp bunları toplamsal paylara dönüştürür.
- Bob'un bitlerini direkt vermeyip aşağıdaki encoding scheme ile maskeleriz.

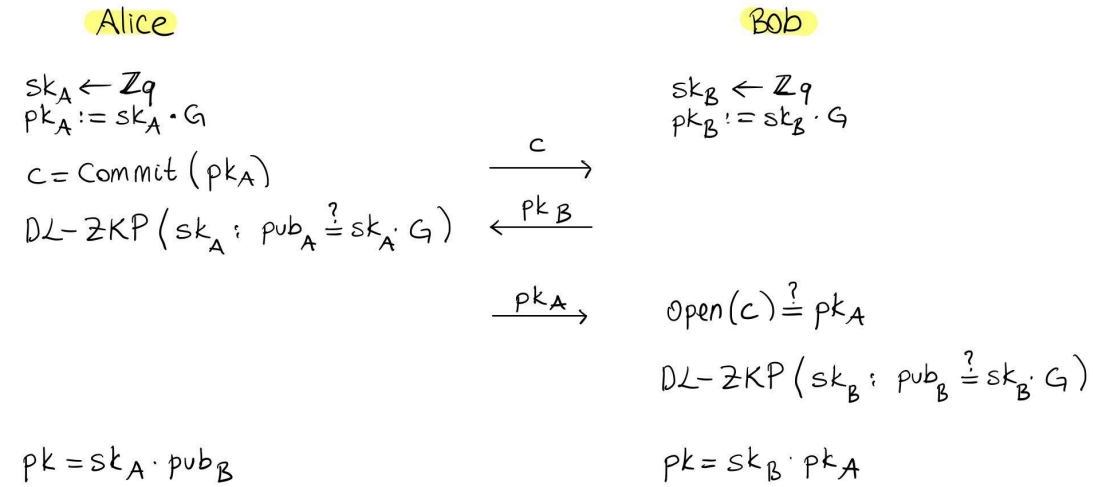
Algoritma: Encode($g^R \in \mathbb{Z}_q^{\kappa+2s}, \beta \in \mathbb{Z}_q$)

1) $g^R \leftarrow \mathbb{Z}_q^{\kappa+2s}$: Public rastgele vektör
2) $\gamma \leftarrow \{0,1\}^{\kappa+2s}$
3) Output Bits($\beta - \langle g^R, \gamma \rangle$)$\|\gamma$

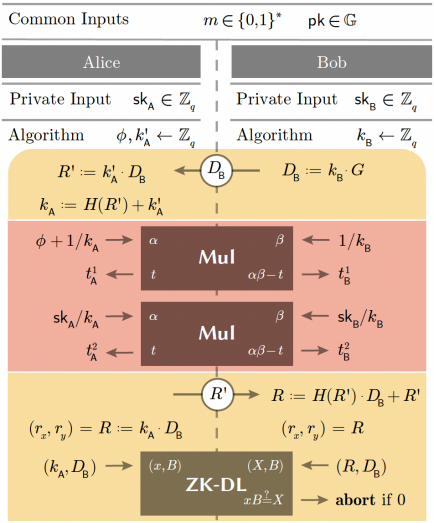
Adım	Alice Girdi: $\alpha \in \mathbb{Z}_q$	Bob Girdi: $\beta \in \mathbb{Z}_q$
1	Rastgele $\hat{\alpha} \leftarrow \mathbb{Z}_q$ seçer. $\boldsymbol{\alpha} = \{\alpha \| \hat{\alpha}\}_{j \in [1, 2\kappa+2s]}$	$\boldsymbol{\omega} = \mathsf{Encode}(g^R, \beta)$ hesaplar.
2	$g^G = \{2^{i-1}\}_{i \in [1,\kappa]} \quad g = g^G \| g^R$: Katsayı vektörü	
3	Alice gönderici olarak $\boldsymbol{\alpha}$, Bob alıcı olarak $\boldsymbol{\omega}$ değerlerini $\mathcal{F}_{\mathsf{COTe}}^{\ell}$'ye gönderir. Her iki taraf aşağıdaki değerleri alır: Alice: $\left\{\mathbf{t}_A \| \hat{\mathbf{t}}_A\right\}_{j \in [1, 2\kappa+2s]}$ Bob: $\left\{\mathbf{t}_B \| \hat{\mathbf{t}}_B\right\}_{j \in [1, 2\kappa+2s]}$	
4	Ortak rasgelelik elde edilir: $\chi, \hat{\chi} \leftarrow H^2(\text{transcript})$	

5	Her j için: $$r_j = \chi \cdot \mathbf{t}_{A_j} + \hat{\chi} \cdot \hat{\mathbf{t}}_{A_j}$$ ve $$u = \chi \cdot a + \hat{\chi} \cdot \hat{\alpha}$$ hesaplayarak $(u, \{r_j\})$'yi Bob'a gönderir.	
6		Bob, her j için doğrulama yapar: $$v_j \cdot \mathbf{t}_{B_j} + \hat{\chi} \cdot \hat{\mathbf{t}}_{B_j} \stackrel{?}{=} \omega_j - u \cdot r_j$$ Eşitlik sağlanmazsa protokolü reddeder.
7	Sonuç: $$t_A = \sum_j g_j \cdot \mathbf{t}_{A_j}$$	Sonuç: $$t_B = \sum_j g_j \cdot \mathbf{t}_{B_j}$$

(2,2)-ECDSA Setup



(2,2)-ECDSA Signing



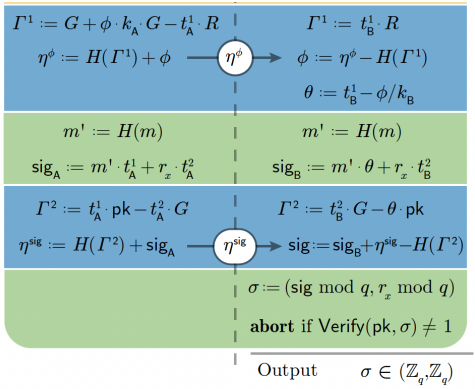
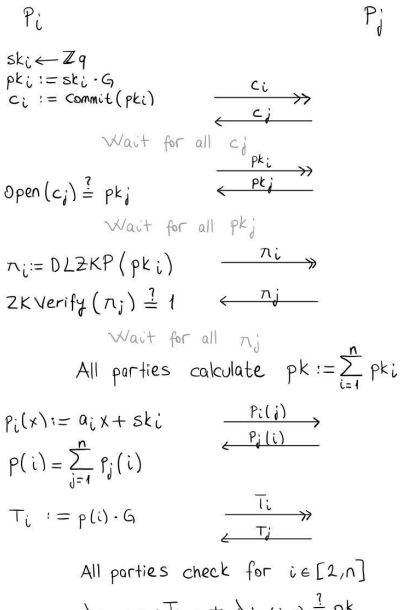
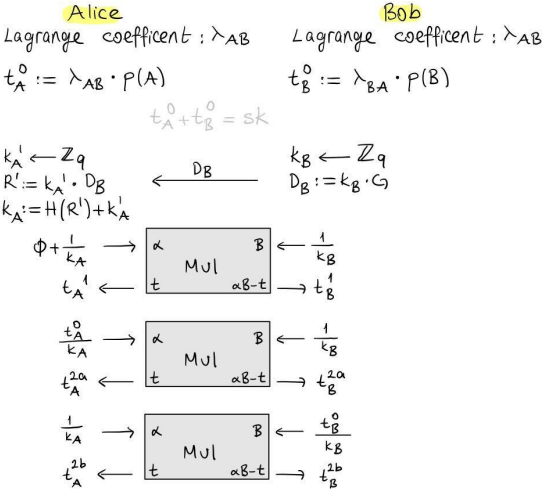


Fig. 1: **Illustrated Two-party Signing Scheme**. Operations are color-coded according to the logical component with which they are associated: Multiplication, Instance Key Exchange, Consistency Check, and Verification/Signing. We specify how to instantiate the multiplication subprotocol (π_{Mul}) in Section VI-B.

(2,n)-ECDSA Setup



(2,n)-ECDSA Signing



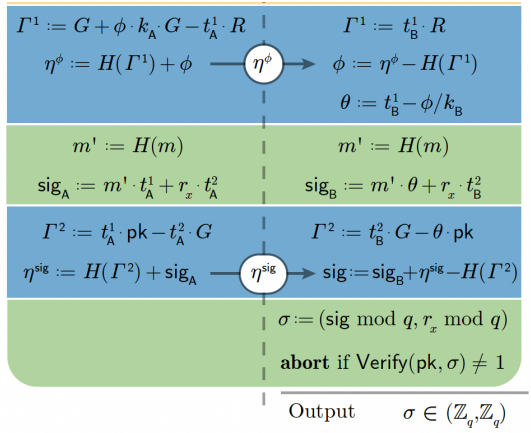


Fig. 1: **Illustrated Two-party Signing Scheme**. Operations are color-coded according to the logical component with which they are associated: Multiplication, Instance Key Exchange, Consistency Check, and Verification/Signing. We specify how to instantiate the multiplication subprotocol (π_{Mul}) in Section VI-B.