



# Enhancing Permissioned Blockchains with Controlled Data Authorization

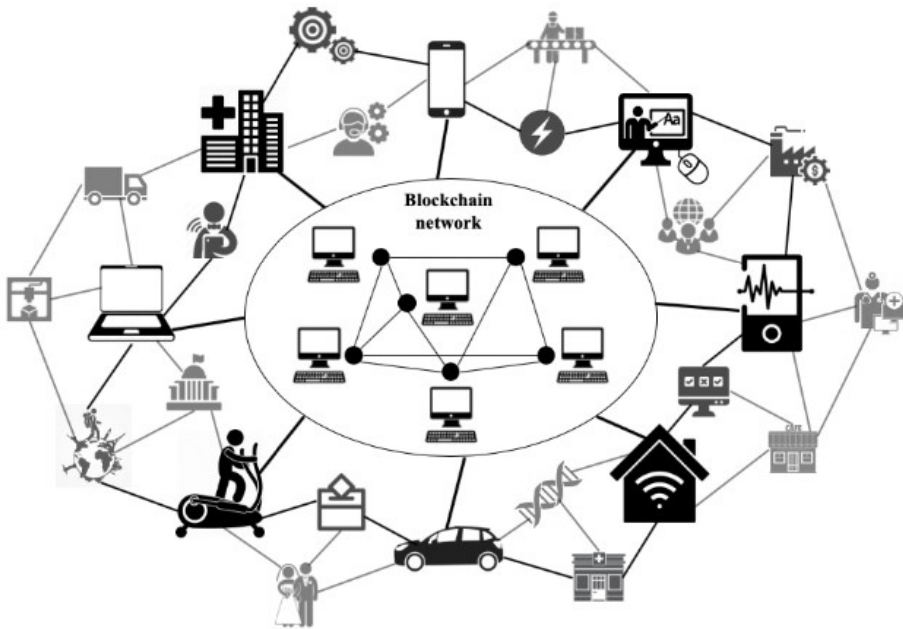
**Qichang Liu**, Xufeng Zhang, Sisi Duan, Haibing Zhang

July 17 2024

# Contents

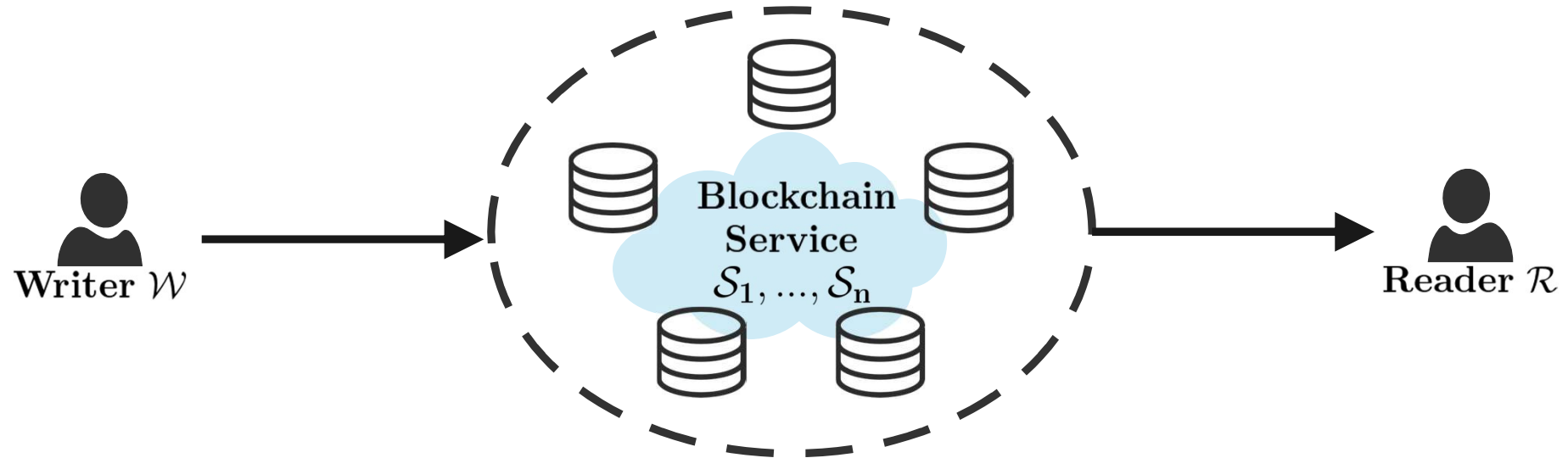
- 1 **Introduction & Motivation**
- 2 The Framework of ECA
- 3 Construction of ECA
- 4 Conclusion

# Goals of permissioned blockchains

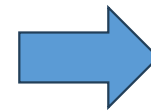


- Data consistency ✓
- Data availability ✓
- Data confidentiality ?

# Confidentiality goals in permissioned blockchains



- Causality preservation
- Confidentiality with access control



Threshold encryption with  
controlled authorization (ECA)







# Contribution

---

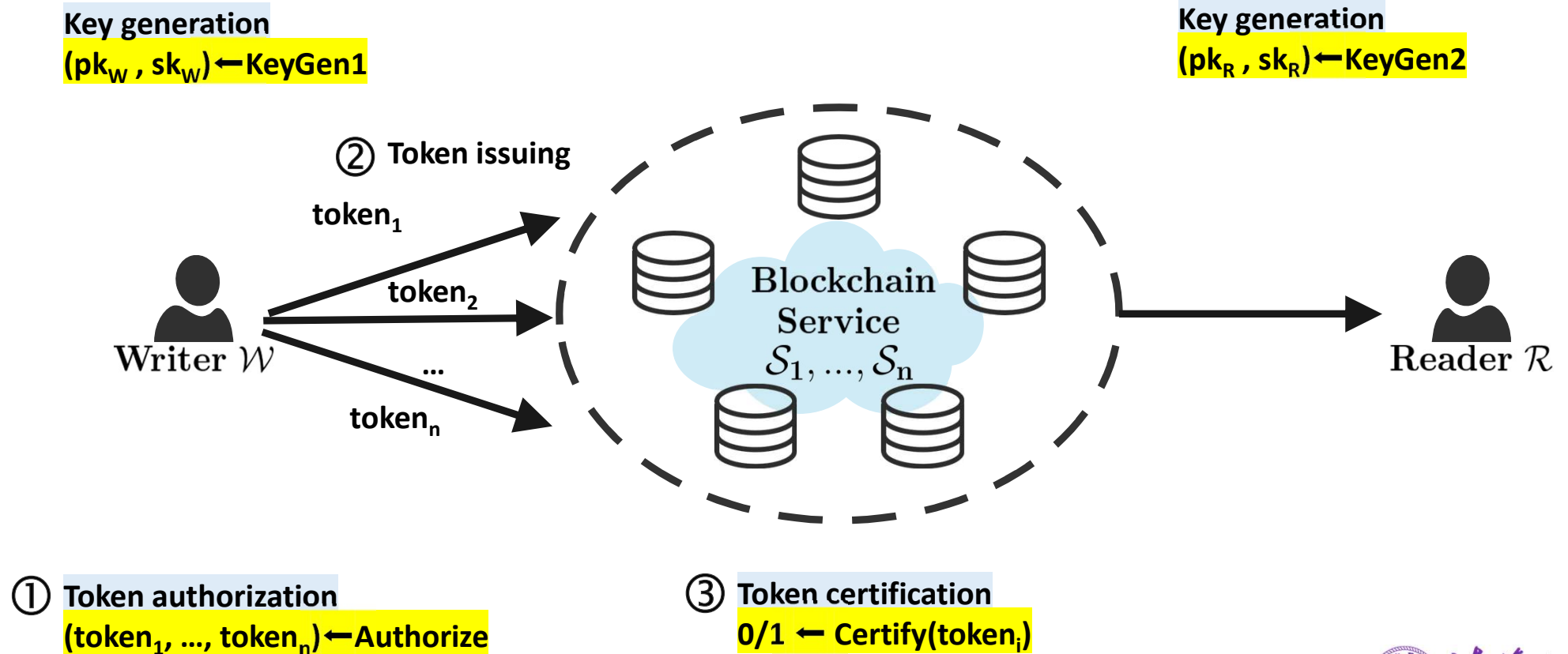
- Defined the primitive of threshold encryption with controlled authorization (ECA), along with three security notions.
- Provided a novel ECA scheme satisfying the security notions we define.
- Developed formal security proofs for our ECA scheme with a novel security reduction to the square Decisional Diffie-Hellman (DDH) assumption.



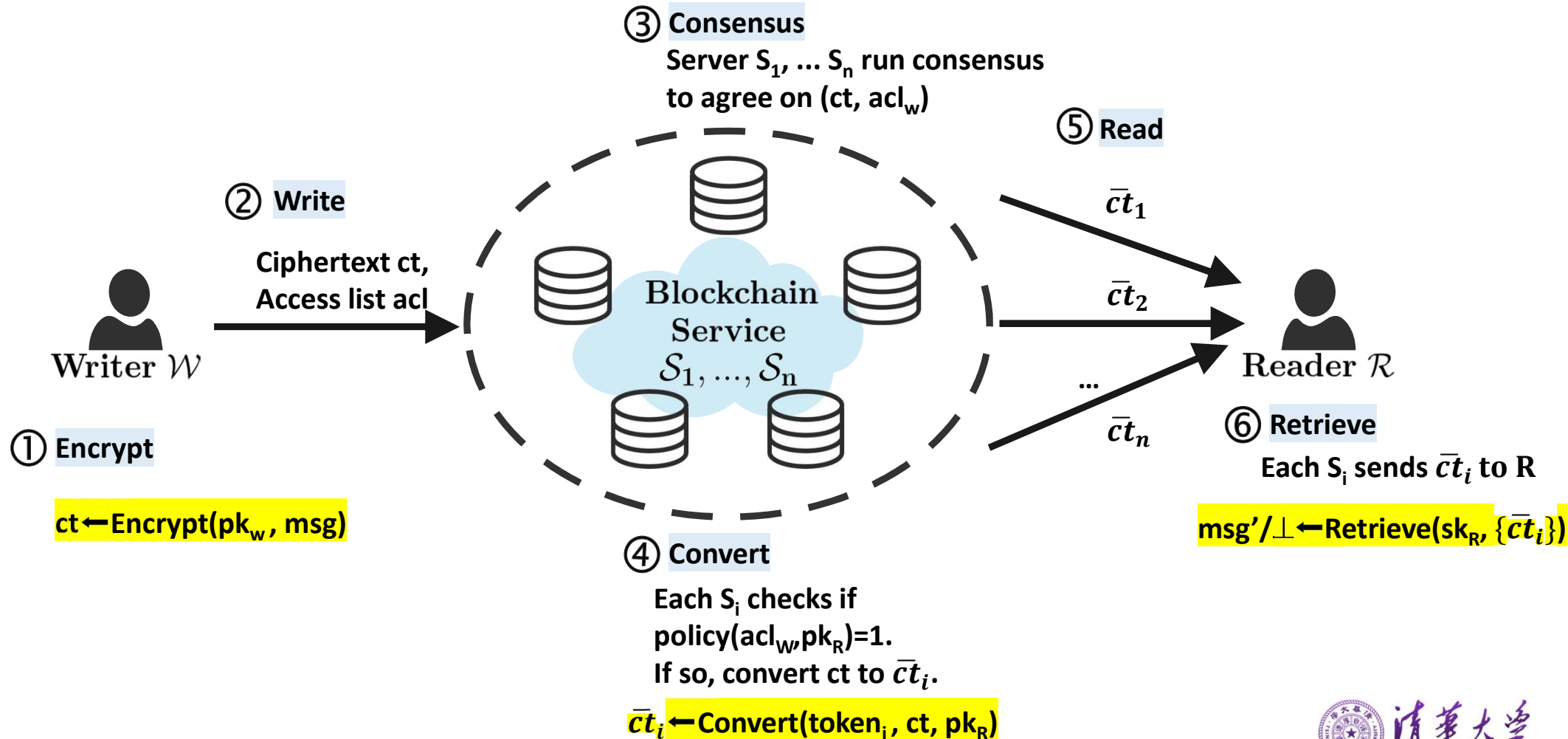
# Contents

- 1 Introduction & Motivation
- 2 **The Framework of ECA**
- 3 Construction of ECA
- 4 Conclusion

# ECA: Authorization



# ECA: Encrypted data transmission



# ECA: Threshold Encryption with Controlled Authorization

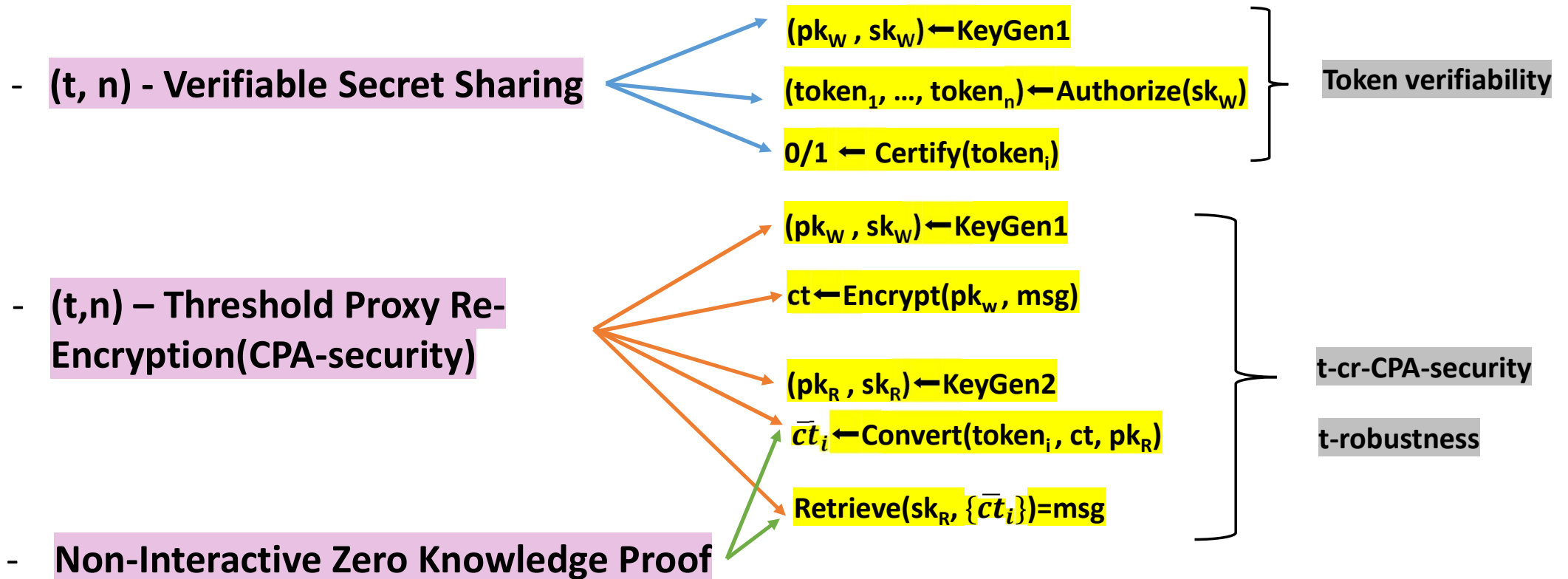
ECA = (KeyGen1, KeyGen2, Authorize, Certify, Encrypt, Convert, Retrieve)

- $(pk_w, sk_w) \leftarrow \text{KeyGen1}$
- $(pk_R, sk_R) \leftarrow \text{KeyGen2}$
- $(token_1, \dots, token_n) \leftarrow \text{Authorize}(sk_w)$
- $0/1 \leftarrow \text{Certify}(token_i)$
- $ct \leftarrow \text{Encrypt}(pk_w, msg)$
- $\bar{ct}_i \leftarrow \text{Convert}(token_i, ct, pk_R)$
- $msg' / \perp \leftarrow \text{Retrieve}(sk_R, \{\bar{ct}_i\})$
- **Token verifiability**  
 $(token_1, \dots, token_n) \leftarrow \text{Authorize}(sk_w)$   
 $1 \leftarrow \text{Certify}(token_i)$
- **t-robustness**  
 $ct \leftarrow \text{Encrypt}(pk_w, msg)$   
 $\bar{ct}_i \leftarrow \text{Convert}(token_i, ct, pk_R)$   
 $\text{Retrieve}(sk_R, \{\bar{ct}_i\}) = msg$   
If at most  $t$  fake  $\bar{ct}_i$ s
- **t-cr-CPA security**  
 $ct \leftarrow \text{Encrypt}(pk_w, msg)$   
The computational indistinguishability of  $ct$  and  $\bar{ct}_i$  for any two plaintexts of the adversary's choice, even if the adversary corrupts at most  $t$  servers by obtaining their tokens

# Contents

- 1 Introduction & Motivation
- 2 The Framework of ECA
- 3 **Construction of ECA**
- 4 Conclusion

# How to achieve ECA?



# Instantiation of the ECA system

## - (t, n) - Verifiable Secret Sharing over a bilinear group

$(pk_{\mathcal{W}}, sk_{\mathcal{W}}) \leftarrow \text{KeyGen1}(\text{pp}):$   
 $\alpha \leftarrow \mathbb{Z}_p, v_0 := g_T^\alpha.$   
 For  $j \in [t]: r_j \leftarrow \mathbb{Z}_p, v_j := g_T^{r_j}.$   
 $sk_{\mathcal{W}} := (\alpha, r_1, \dots, r_t).$   
 $r(x) := \alpha + \sum_{j=1}^t r_j \cdot x^j \pmod p.$   
 For  $i \in [n]:$   
 $\theta_i := r(i), w_i = e(g_1^{1/\alpha}, g_2^{\theta_i}).$   
 $pk_{\mathcal{W}} := (g_1^{1/\alpha}, (v_0, v_1, \dots, v_t), (w_1, \dots, w_n)).$   
 Return  $(pk_{\mathcal{W}}, sk_{\mathcal{W}}).$

**Verifiable Property**

$(\text{token}_1, \dots, \text{token}_n) \leftarrow \text{Authorize}(sk_{\mathcal{W}}):$   
 Parse  $sk_{\mathcal{W}} = (\alpha, r_1, \dots, r_t).$   
 $r(x) := \alpha + \sum_{j=1}^t r_j \cdot x^j \pmod p.$   
 For  $i \in [n]:$   
 $\theta_i := r(i).$   
 $\text{token}_i := \theta_i.$   
 Return  $(\text{token}_1, \dots, \text{token}_n).$

**(t,n)-Secret Sharing**

$0/1 \leftarrow \text{Certify}(pk_{\mathcal{W}}, i, \text{token}_i):$  // validity of tokens  
 Parse  $pk_{\mathcal{W}} = (v', (v_0, v_1, \dots, v_t), (w_1, w_2, \dots, w_n)).$   
 Parse  $\text{token}_i = \theta_i.$   
 //test consistency between  $\theta_i, w_i$  and  $(v', v_0, \dots, v_t)$   
 If  $(g_T^{\theta_i} \neq \prod_{\ell=0}^t v_\ell^{i^\ell}) \vee (e(v', g_2^{\theta_i}) \neq w_i):$   
 Return 0. //token invalid  
 For  $k \in [t+1], j \in [t+2, n]:$   
 $\lambda_{kj} := \prod_{i=1, i \neq k}^n \frac{j-i}{k-i}.$  //Lagrange coefficients  
 //test consistency between  $v'$  and  $(w_1, \dots, w_{t+1})$   
 If  $(\prod_{k=1}^{t+1} w_k^{\lambda_{k0}} \neq g_T):$  Return 0; //token invalid  
 //test consistency of  $(w_1, \dots, w_n)$   
 For  $j \in [t+2, n]$   
 If  $\prod_{k=1}^{t+1} w_k^{\lambda_{kj}} \neq w_j:$  Return 0; //token invalid  
 Else: Return 1. //token valid

**Verifiable Property**

# Instantiation of the ECA system

## - (t,n)-Threshold Proxy Re-Encryption

$(ct, acl_W) \leftarrow \text{Encrypt}(pk_W, msg, acl_W):$   
 Parse  $pk_W = (v', (v_0, v_1, \dots, v_t), (w_1, w_2, \dots, w_n))$ .  
 $r \leftarrow \$ \mathbb{Z}_p$ ,  $ct_1 := v'^r$ ,  $k := g_T^r$ .  
 $ct_2 \leftarrow \text{SE.Enc}(k, msg)$ .  
 Return  $(ct = (ct_1, ct_2), acl_W)$ .

$(pk_R, sk_R) \leftarrow \text{KeyGen2}(pp):$   
 $\beta \leftarrow \$ \mathbb{Z}_p$ ,  $sk_R := \beta$ ,  $pk_R := g_2^\beta$ .  
 Return  $(pk_R, sk_R)$ .

**Proof Generation: Prove that  $\bar{ct}_i$   
 is generated by the correct  
 token  $\theta_i$**

## - Non-Interactive Zero Knowledge Proof

$\bar{ct}_i \leftarrow \text{Convert}(\text{token}_i, (pk_W, ct, acl_W), pk_R = g_2^\beta, \text{policy}):$   
 If  $\text{policy}(acl_W, pk_R) = 0$ : Return  $\perp$ .  
 Parse  $\text{token}_i = \theta_i$ .  
 Parse  $ct = (ct_1, ct_2)$ .  
 $h := e(ct_1, pk_R) = e(ct_1, g_2^\beta)$ ,  $\bar{h}_i := h^{\theta_i}$ .  
 $// \pi_i \leftarrow \text{ZKProof}((h, \bar{h}_i, g_T, \bar{u}_i), \theta_i)$   
 $\bar{u}_i := g_T^{\theta_i}$ .  
 $s_i \leftarrow \$ \mathbb{Z}_p$ ,  $\hat{h}_i := h^{s_i}$ ,  $\hat{u}_i := g_T^{s_i}$ .  
 $e_i := H(h, g_T, \bar{h}_i, \bar{u}_i, \hat{h}_i, \hat{u}_i)$ .  
 $f_i := s_i + \theta_i \cdot e_i \mod p$ .  
 $\pi_i := (\hat{h}_i, \hat{u}_i, f_i)$ .  $// \text{proof of } \log_h \bar{h}_i = \log_{g_T} \bar{u}_i$   
 $\bar{ct}_{i,0} := h$ .  
 $\bar{ct}_{i,1} := (\bar{h}_i, \pi_i)$ ,  $\bar{ct}_{i,2} := ct_2$ .  
 Return  $\bar{ct}_i = (\bar{ct}_{i,0}, \bar{ct}_{i,1}, \bar{ct}_{i,2})$ .



# Instantiation of the ECA system

## - (t,n)-Threshold Proxy Re-Encryption

$\bar{ct}_i \leftarrow \text{Convert}(\text{token}_i, (pk_W, ct, acl_W), pk_R = g_2^\beta, \text{policy}):$   
 If  $\text{policy}(acl_W, pk_R) = 0$ : Return  $\perp$ .  
 Parse  $\text{token}_i = \theta_i$ .  
 Parse  $ct = (ct_1, ct_2)$ .  
 $h := e(ct_1, pk_R) = e(ct_1, g_2^\beta)$ ,  $\bar{h}_i := h^{\theta_i}$ .  
 $// \pi_i \leftarrow \text{ZKProof}((h, \bar{h}_i, g_T, \bar{u}_i), \theta_i)$   
 $\bar{u}_i := g_T^{\theta_i}$ .  
 $s_i \leftarrow \mathbb{Z}_p$ ,  $\hat{h}_i := h^{s_i}$ ,  $\hat{u}_i := g_T^{s_i}$ .  
 $e_i := H(h, g_T, \bar{h}_i, \bar{u}_i, \hat{h}_i, \hat{u}_i)$ .  
 $f_i := s_i + \theta_i \cdot e_i \pmod p$ .  
 $\pi_i := (\bar{h}_i, \hat{u}_i, f_i)$ .  
 $\bar{ct}_{i,0} := h$ .  
 $\bar{ct}_{i,1} := (\bar{h}_i, \pi_i)$ ,  $\bar{ct}_{i,2} := \dots$ .  
 Return  $\bar{ct}_i = (\bar{ct}_{i,0}, \bar{ct}_{i,1}, \bar{ct}_{i,2})$ .

**Proof Generation: Prove that  $\bar{ct}_i$  is generated by the correct token  $\theta_i$**

Q. Liu, X. Zhang, J. Du, H. Zhang

## - Noninteractive Zero Knowledge Proof

$\text{msg}'/\perp \leftarrow \text{Retrieve}(sk_R = \beta, \{(\ell_j, \bar{ct}_{\ell_j})\}_{j \in [n']}):$   
 If  $n' \leq t$ : Return  $\perp$ .  
 Parse  $\bar{ct}_{\ell_j} = (\bar{ct}_{\ell_j,0}, \bar{ct}_{\ell_j,1}, \bar{ct}_{\ell_j,2})$  for  $j \in [n']$ .  
 Filter set  $\{(\ell_j, \bar{ct}_{\ell_j})\}_{j \in [n']}$  to obtain subset  $\mathcal{S} = \{(i_j, \bar{ct}_{i_j})\}_{j \in [n']}$  satisfying  $n'' \geq t + 1$   
 $\bigwedge_{j, j' \in [n'']} ((\bar{ct}_{i_j,0} = \bar{ct}_{i_{j'},0}) \wedge (\bar{ct}_{i_j,2} = \bar{ct}_{i_{j'},2})) : \quad (\star)$   
 $\mathcal{V} := \emptyset$ .  
 For  $j \in [n'']$   
 Parse  $\bar{ct}_{i_j,0} = h$ ,  $\bar{ct}_{i_j,1} = (\bar{h}_{i_j}, \pi_{i_j} = (\hat{h}_{i_j}, \hat{u}_{i_j}, f_{i_j}))$ .  
 $// 0/1 \leftarrow \text{ZKVerify}((h, \bar{h}_{i_j}, g_T, \bar{u}_{i_j}), \pi_{i_j})$   
 $\bar{u}_{i_j}' := \prod_{\ell=0}^t (v_\ell)^{i_{j,\ell}}$ .  
 $e_{i_j} := H(h, g_T, \bar{h}_{i_j}, \bar{u}_{i_j}', \hat{h}_{i_j}, \hat{u}_{i_j})$ .  
 If  $(h^{f_{i_j}} = \hat{h}_{i_j} \cdot \bar{h}_{i_j}^{e_{i_j}}) \wedge (g_T^{f_{i_j}} = \hat{u}_{i_j} \cdot \bar{u}_{i_j}'^{e_{i_j}}): \quad (\star\star)$   
 $\mathcal{V} := \mathcal{V} \cup \{(i_j, \bar{h}_{i_j})\}$ .  
 $// \text{end of proof verification of } \log_h h_{i_j} = \log_{g_T} \bar{u}_{i_j}'$   
 If  $|\mathcal{V}| \leq t$ : Return  $\perp$ .  
 Parse  $\mathcal{V} = \{(j_0, \bar{h}_{j_0}), (j_1, \bar{h}_{j_1}), \dots, (j_t, \bar{h}_{j_t}), \dots\}$ .  
 $k' := \left( \prod_{i=0}^t \bar{h}_{j_i}^{\lambda_{j_i,0}} \right)^{\beta^{-1}}$ .  $// \lambda_{j_i,0}$ 's are Lagrange coefficients  
 $\text{msg}' \leftarrow \text{SE.Dec}(k', \bar{ct}_{j_1,2})$ .  $// \bar{ct}_{j_1,2} = \bar{ct}_{j_2,2} = \dots = \bar{ct}_{j_t,2}$   
 Return  $\text{msg}'$ .

**Proof Verification: Verify that  $\bar{ct}_i$  is generated by the correct token  $\theta_i$**

# A quick recap

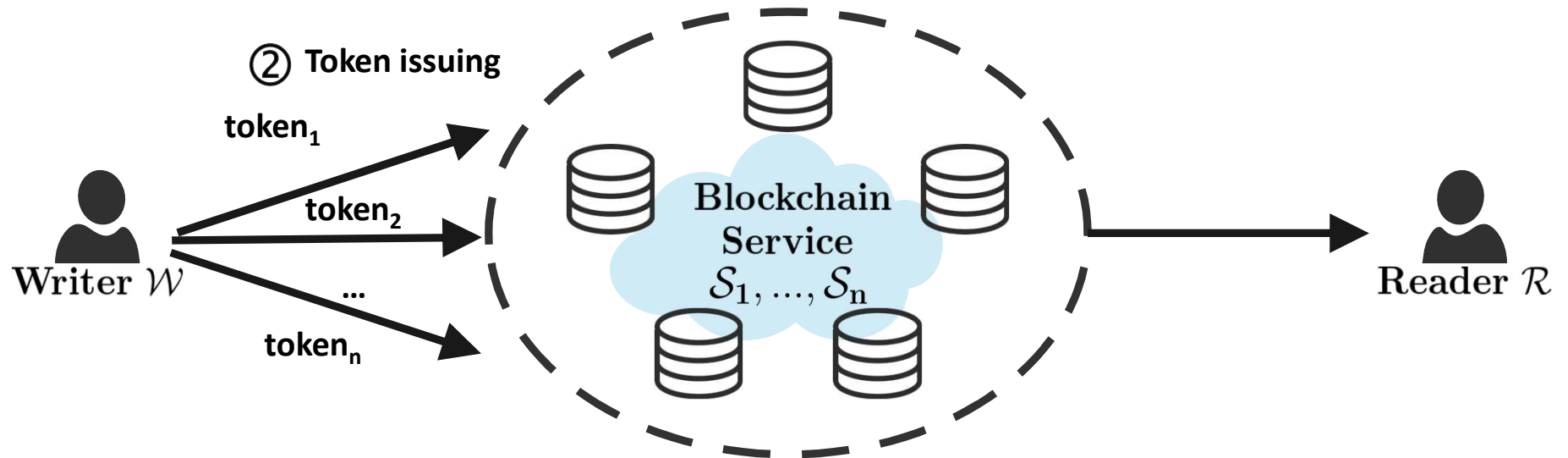
## - (t, n) - Verifiable Secret Sharing

Key generation

$(pk_W, sk_W) \leftarrow \text{KeyGen1}$

Key generation

$(pk_R, sk_R) \leftarrow \text{KeyGen2}$



① Token authorization

$(\text{token}_1, \dots, \text{token}_n) \leftarrow \text{Authorize}$

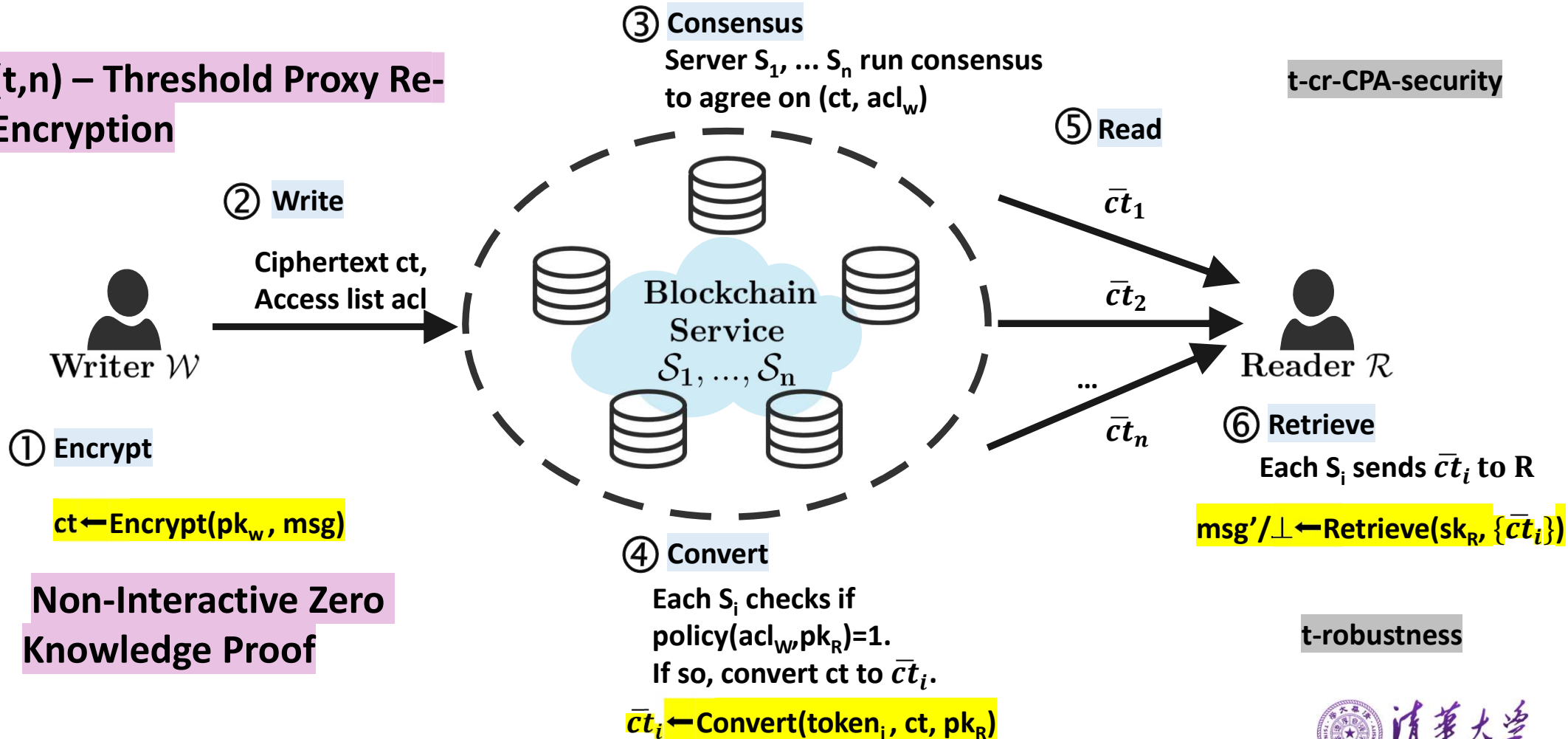
③ Token certification

$0/1 \leftarrow \text{Certify}(\text{token}_i)$

Token verifiability

# A quick recap

## - (t,n) – Threshold Proxy Re-Encryption



# Contents

- 1 Introduction & Motivation
- 2 The Framework of ECA
- 3 Construction of ECA
- 4 **Conclusion**

# Conclusion

---

## **Confidentiality of blockchains enabled by ECA:**

- ECA ( $t$ -robustness): Reader's successful decryption in case of  $t$  corrupted replicas.
- ECA (Token verifiability): Replicas are able to verify tokens distributed by the writer, writers cannot cheat replicas.
- ECA ( $t$ -cr-CPA security): Confidentiality will be preserved in case of  $t$  colluding replicas.
- Causality preservation & Flexible token revocation

**Thank you!**