K-Time_Modifiable_and_Epoch-Based_Redactable_Blockchain

Block chain Introduction

- Widely used in digital currency, supply chain, insurance, agency…
- immutable appendonly ledger
- Relies on a hash chain that links multiple blocks together (Usually)
- Allow forks (most blockchain system)
- immutability property : 區塊鏈發展障礙

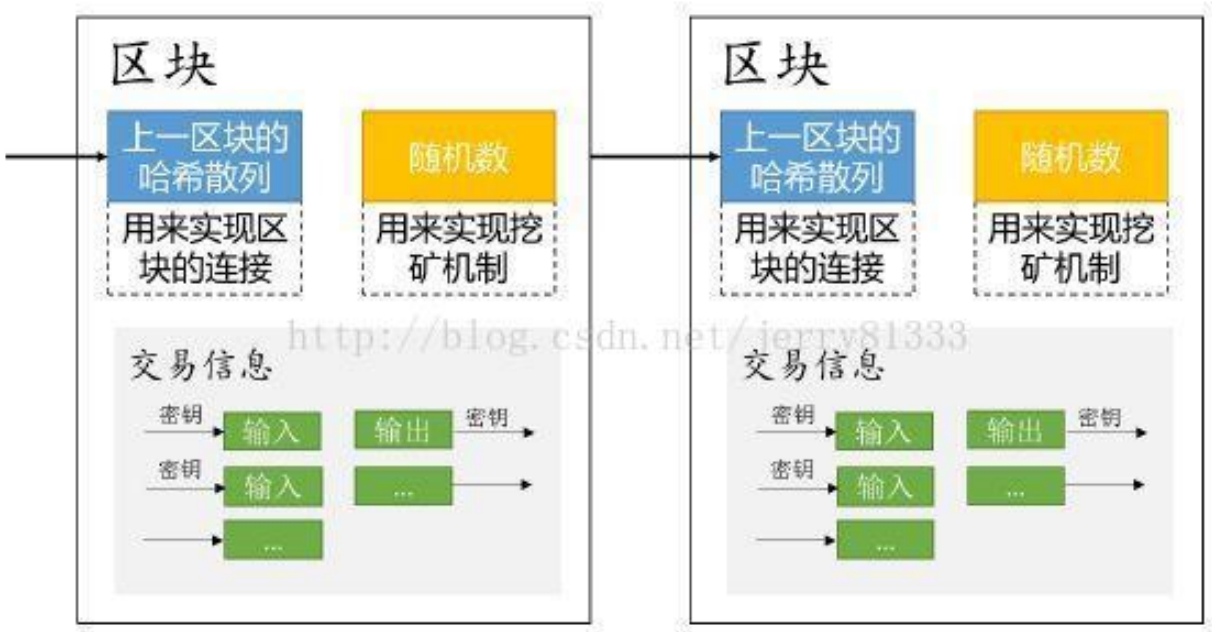
Blockchain 不可更改性

- 被濫用於散播不恰當內容 (e.g., child pornography and material that infringes on intellectual rights)
- The Right to be Forgotten (個人隱私資訊 : 歐盟2006 GDPR)

- Solution ?

Related Work

- μ chain by Puddu (DPSS) [22]
- Novel data erasing approach by Florian [23]
- the concept of redactable blockchain was first introduced by Ateniese [24]



Related Work

- CHET and ABE by Derler [25]
- To eliminate the trusted CA :
  - Deuber *et al.* [26] proposed a block-level redactable blockchain
  - Dishonest and rational miners may not check the candidate block pool for more opportunities to be a new block finder
  - Vulnerable to denial-of-service attacks if the cost of applying for an amendment is not high enough

Contributions

- *k*-time modifiable and epoch-based redactable blockchain (KERB)
  - *K-time modification*
  - *Epoch-based redaction*
  - *Strong security model*

COMPARISON AMONG CURRENT REDACTABLE BLOCKCHAIN SOLUTIONS AND OURS

	Type	***Core Technology***	***Authority***	***Modifier***	***Requirement***
AMVA17 [24]	*Permissioned*	*PKI, CH*	*CA*	*Authorized Users*	*Secret Key*
DSSS19 [25]	*Permissioned*	*ABE, CHET*	*CA*	*Authorized Users*	*Secret Key*
DMT19 [26]	*Permissionless*	*Consensus-Based Voting*	*N/A*	*All Miners*	*Pass Voting*
Ours	*Permissioned*	*Signatures, CH*	*CA*	*Authorized Users*	*Token Key*

Contributions

	K-Time	*Epoch-Based*	*Penalty*	*Accountability*	*Transparency*	*Collusion*
AMVA17 [24]	✗	✗	✗	✓	✗	✓
DSSS19 [25]	✗	✗	✗	✗	✗	✗
DMT19 [26]	✗	✗	✗	✓	✓	*N/A*
Ours	✓	✓	✓	✓	✓	✓

- "***K-Time***" is short for k-time modification that means an authorized user is allowed to operate rewriting at most k times (in an epoch).
- "***Epoch-Based***" is short for epoch-based modification that means the validation of modification privilege is epoch-based for revoking compromised and invalid modifiers.
- "***Penalty***" is short for monetary penalty for penalizing malicious modifiers.
- "***Accountability***" means the identity of modifiers can be identified.
- "***Transparency***" means the transaction can be identified from the chain participants or modifiers.
- "***Collusion***" is short for collusion attack that means the chain participants and modifiers who have not a specific authorization collusion to launch modification operations.

Advantage

- more generic chameleon hashes rather than enhanced collision-resistance [24], and even CHET [25].
- no trapdoor correctness checking requirement
- We apply digital signatures to endorse rewriting privileges rather than encryption mechanisms for optimal performance
- suitable for a large-scale setting

Challenges

Functionality

- *K*-time modification is helpful to reduce the loss when key leakage or faulty operation happens
- Epoch-based modification limits the validation period of rewriting privileges (Quite challenging : ABE(RABE)?)
- ABE
  - semi-trusted third party ?
  - Costly to synchronize
  - Easy to find historical information

Security

- CHET
  - Two trapdoors (by CA and by transaction owner)

Technical Overview

- To eliminate the costly encryption mechanism, we apply digital signatures to release the trapdoor of chameleon hashes.
  - only the rewriting from the authorized modifier can pass the verification
- Double-authentication-preventing signatures
- Messages consist of an address and a payload component, and a signer is penalized if two messages with the same addresses but different payloads are signed
- Schnorr signature
  - improve efficiency
  - rather than costly zero-knowledge proof Systems

Technical Overview

- reduce the computational cost
  - 1-degree polynomial to realize secret sharing
- “{Admin, Finance}” : “Admin AND Finance”
  - cannot rewrite the transaction specified by a policy “Admin AND Healthcare”
- CA issues the rewriting privileges by signing the attribute set of the modifier (and the public key of this modifier). If a transaction associated access policy complies with the attribute set, the modifier can rewrite this transaction

Digital Signatures

- *A digital signature DS with a message space M consists of four algorithms {* Setup, KeyGen, Sign, Verify *}*

$\mathcal{DS}.\mathsf{Setup}(1^\lambda) \rightarrow pp$: *On input a security parameter* $\lambda \in \mathbb{N}$, *and output a public parameter pp, where pp is implicit input to all other algorithms.*

Digital Signatures

$\mathcal{DS}.\mathsf{KeyGen}(pp) \rightarrow (sk, pk)$: *On input a public parameter* pp, *and output a secret key* sk *and a public key* pk.

$\mathcal{DS}.\mathsf{Sign}(sk, m) \rightarrow \sigma$: *On input a secret key* sk *and a message* $m \in \mathcal{M}$, *and output a signature* σ.

$\mathcal{DS}.\mathsf{Verify}(pk, \sigma, m) \rightarrow b$: *On input a public key* pk, *a signature* σ *and a message* $m \in \mathcal{M}$, *and output a decision bit* $b \in \{0, 1\}$.

Definition 2 (Correctness): A digital signature $\mathcal{DS}$ *is called correct, if for all security parameters* $\lambda \in \mathbb{N}$, *for all* $pp \leftarrow \mathcal{DS}.\mathsf{Setup}(1^\lambda)$, *for all* $(sk, pk) \leftarrow \mathcal{DS}.\mathsf{KeyGen}(pp)$, *for all* $m \in \mathcal{M}$, $\mathcal{DS}.\mathsf{Verify}(pk, \mathcal{DS}.\mathsf{Sign}(sk, m), m) = 1$ *is always true.*

Digital Signatures

*Definition 3 (*EUF-CMA *Security): The* EUF-CMA *security definition of a digital signature $\mathcal{DS}$ is based on the following experiment.*

$\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathcal{DS}}(1^\lambda)$
$pp \leftarrow \mathcal{DS}.\mathsf{Setup}(1^\lambda)$;
$\mathcal{L}_{\mathsf{key}} \leftarrow \emptyset$; //KeyGen *query list*
$\mathcal{L}_{\mathsf{sign}} \leftarrow \emptyset$; //Sign *query list*
$\mathcal{L}_{\mathsf{corr}} \leftarrow \emptyset$; //Corrupt *query list*
$(pk^*, m^*, \sigma^*) \leftarrow$
$\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}}(\cdot),\mathcal{O}_{\mathsf{Sign}},\mathcal{O}_{\mathsf{Corrupt}}(\cdot)}(pp)$;
return 1 if $pk^* \notin \mathcal{L}_{\mathsf{corr}} \wedge$
$(pk^*, m^*) \notin \mathcal{L}_{\mathsf{sign}} \wedge$
$\mathcal{DS}.\mathsf{Verify}(pk^*, \sigma^*, m^*) = 1$,
else return 0.

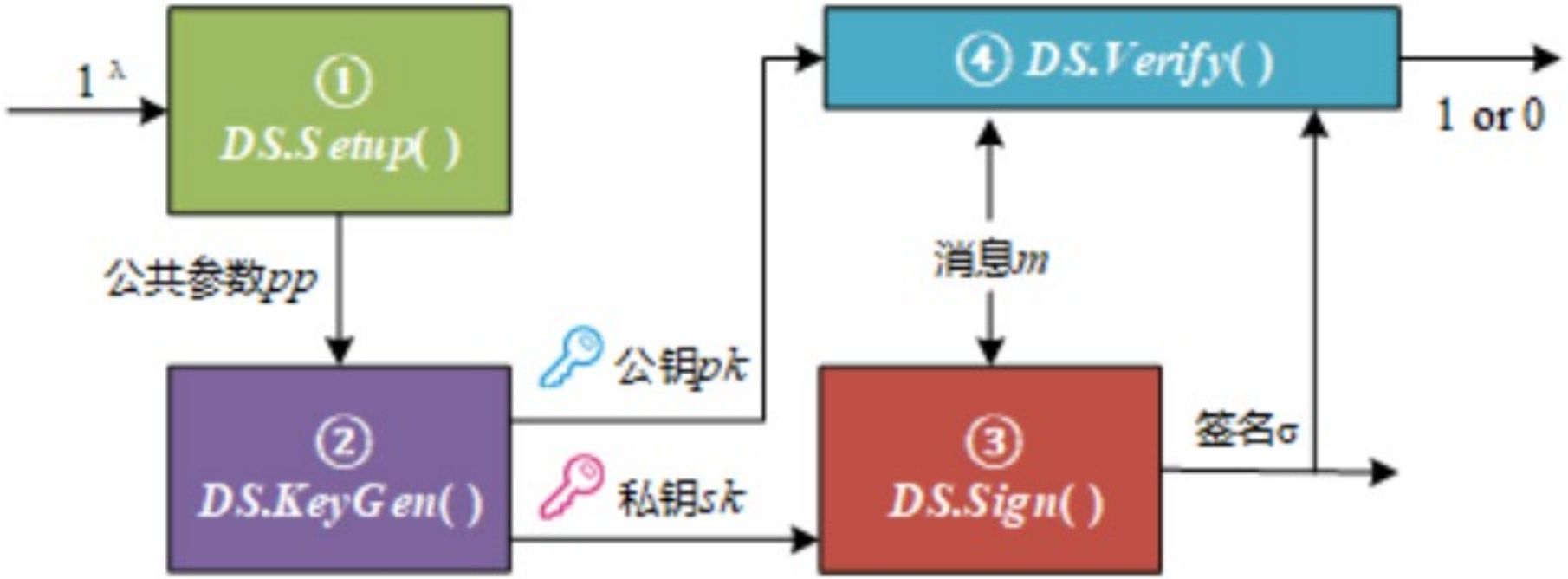
Oracle $\mathcal{O}_{\mathsf{KeyGen}}(i)$
$(sk, pk) \leftarrow \mathcal{DS}.\mathsf{KeyGen}(pp)$;
$\mathcal{L}_{\mathsf{key}} \leftarrow \mathcal{L}_{\mathsf{key}} \cup \{(i, sk, pk)\}$;
return pk.

Oracle $\mathcal{O}_{\mathsf{Sign}}(pk, m)$
$\sigma \leftarrow \mathcal{DS}.\mathsf{Sign}(sk, m)$;
$\mathcal{L}_{\mathsf{sign}} \leftarrow \mathcal{L}_{\mathsf{sign}} \cup \{(pk, m)\}$;
return σ.

Oracle $\mathcal{O}_{\mathsf{Corrupt}}(pk)$
$\mathcal{L}_{\mathsf{corr}} \leftarrow \mathcal{L}_{\mathsf{corr}} \cup \{pk\}$;
return sk.

Digital Signatures

$$\mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{DS}}(1^{\lambda}) = \left| \Pr[\mathbf{Exp}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{DS}}(1^{\lambda}) = 1] \right|.$$



Chameleon Hashes

*Definition 4 (Chameleon Hashes): A chameleon hash $\mathcal{CH}$ with a message space $\mathcal{M}$ consists of five algorithms {*Setup, KeyGen, Hash, Verify, Adapt*}:*

$\mathcal{CH}$.Setup$(1^\lambda) \rightarrow pp$: *On input a security parameter $\lambda \in \mathbb{N}$, and output a public parameter pp, where pp is implicit input to all other algorithms.*

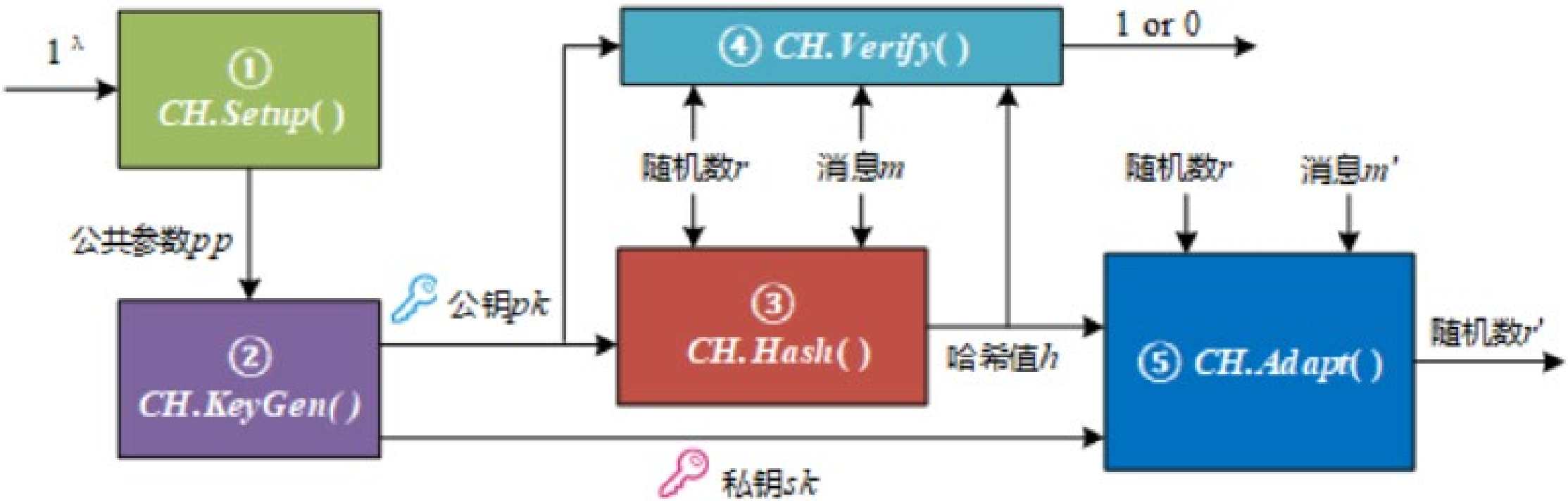
$\mathcal{CH}$.KeyGen$(pp) \rightarrow (sk, pk)$: *On input a public parameter pp, and output a secret key sk and a public key pk.*

$\mathcal{CH}$.Hash$(pk, m; r) \rightarrow h$: *On input a public key pk, a message $m \in \mathcal{M}$ and a randomness r, and output a hash value h.*

$\mathcal{CH}$.Verify$(pk, m, h, r) \rightarrow b$: *On input a public key pk, a message $m \in \mathcal{M}$, a hash value h and a randomness r, and output a decision bit $b \in \{0, 1\}$.*

$\mathcal{CH}$.Adapt$(sk, m, h, r, m') \rightarrow r'$: *On input a secret key sk, a message $m \in \mathcal{M}$, a hash value h, a randomness r and a message $m' \in \mathcal{M}$, and output a randomness r'.*

Chameleon Hashes



K-TIME MODIFIABLE AND EPOCH-BASED REDACTABLE BLOCKCHAIN

Definition 7 (KERB): A k-time modifiable and epoch-based redactable blockchain $\mathcal{KERB}$ involves four types of parties: CA, users, modifiers, and miners. It consists of the following eight algorithms:

$\mathcal{KERB}.\mathsf{Setup}(1^\lambda) \rightarrow (pp, msk, mpk)$: The probabilistic setup algorithm is run by CA. It takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs a public parameter pp, a master secret key msk and a master public key mpk, where pp and mpk are implicit input to all other algorithms.

$\mathcal{KERB}.\mathsf{Setup_m}(pp, n) \rightarrow (sk_m, pk_m)$: The probabilistic modifier setup algorithm is run by each modifier. It takes a public parameter pp and a number $n \in \mathbb{N}$ as input, and outputs a secret key sk_m and a public key pk_m, where n is the maximum number of redaction operations associated with the key pair (sk_m, pk_m).

K-TIME MODIFIABLE AND EPOCH-BASED REDACTABLE BLOCKCHAIN

$\mathcal{KERB}.\mathsf{Setup_u}(pp) \rightarrow (sk_u, pk_u)$: *The probabilistic user setup algorithm is run by each user. It takes a public parameter* pp *as input, and outputs a secret key* sk_u *and a public key* pk_u.

$\mathcal{KERB}.\mathsf{TKGen}(msk, pk_m, k, \mathcal{S}, t) \rightarrow tk$: *The probabilistic token key generation algorithm is run by CA. It takes a master secret key* msk, *a public key* pk_m, *a number* $k \in \mathbb{N}$, *a set of attributes* $\mathcal{S}$ *and a timestamp* t, *and outputs a token key* tk, *where* k *is maximum number of redaction operations for the public key* pk_m *permitted by CA and* t *is the expiration date of the modification privilege.*

K-TIME MODIFIABLE AND EPOCH-BASED REDACTABLE BLOCKCHAIN

$\mathcal{KERB}.\mathsf{Hash}(mpk, sk_u, (\mathsf{ID}, tx_{\mathsf{ID}}), \mathbb{A}) \rightarrow (h, r, \sigma_{\mathsf{ID}})$: *The probabilistic hash algorithm is run by each user. It takes a master public key mpk, a secret key* sk_u*, a message including a transaction identity* ID *and its content* tx_{ID}*, and an access structure* $\mathbb{A}$ *as input, and outputs a hash value h, a randomness r and a signature* σ_{ID}.

$\mathcal{KERB}.\mathsf{Adapt}(mpk, sk_m, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, (\mathsf{ID}, tx'_{\mathsf{ID}})) \rightarrow (r', \sigma'_{\mathsf{ID}})$: *The probabilistic adaption algorithm is run by each modifier. It takes a master public key mpk, a secret key* sk_m*, a message including a transaction identity* ID *and its content* tx_{ID}*, a hash value h, a randomness r, a signature* σ_{ID} *and a message including a transaction identity* ID *and its content* tx'_{ID} *as input, and outputs a randomness* r' *and a signature* σ'_{ID}.

K-TIME MODIFIABLE AND EPOCH-BASED REDACTABLE BLOCKCHAIN

$\mathcal{KERB}.\overline{\mathsf{Verify}}(mpk, pk, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, t') \rightarrow b$: *The deterministic verification algorithm is run by miners. It takes a master public key* mpk, *a public key* pk, *a message including a transaction identity* ID *and its content* tx_{ID}, *a hash value* h, *a randomness* r, *a signature* σ_{ID} *and a current timestamp* t' *as input, and outputs a decision bit* $b \in \{0, 1\}$ *indicating whether the transaction* $(\mathsf{ID}, tx_{\mathsf{ID}})$ *is valid or not, where the input public key* pk *is either the public key of the user* pk_u *(if the message has not been redacted) or the public key of the user and modifier implicitly associated with the token key* (pk_u, pk_m, tk) *(if the message has been redacted).*

K-TIME MODIFIABLE AND EPOCH-BASED REDACTABLE BLOCKCHAIN

$\mathcal{KERB}.\mathsf{Ex}(pk_m, (i, \mathsf{ID}, r, \mathbb{A}), (i', \mathsf{ID}', r', \mathbb{A}'), \sigma_{\mathsf{ID}}, \sigma_{\mathsf{ID}'})$
$\rightarrow$ *sk'_m: The deterministic extraction algorithm is run by any party. It takes a public key pk_m, a message including an index i, a transaction identity ID, a randomness r and an access structure $\mathbb{A}$, a message including an index i', a transaction identity ID', a randomness r' and an access structure $\mathbb{A}'$, and two signatures $(\sigma_{\mathsf{ID}}, \sigma_{\mathsf{ID}'})$ as input. It outputs a secret key sk'_m, where sk'_m is the secret signing key of the modifier.*

Definition 8 (Colliding Redaction): We say that a signature σ_{ID} on the message $(i, \mathsf{ID}, r, \mathbb{A})$ and a signature $\sigma_{\mathsf{ID}'}$ on the message $(i', \mathsf{ID}', r', \mathbb{A}')$ are a colliding redaction if $i = i'$, but $(\mathsf{ID}, r, \mathbb{A}) \neq (\mathsf{ID}', r', \mathbb{A}')$.

K-TIME MODIFIABLE AND EPOCH-BASED REDACTABLE BLOCKCHAIN

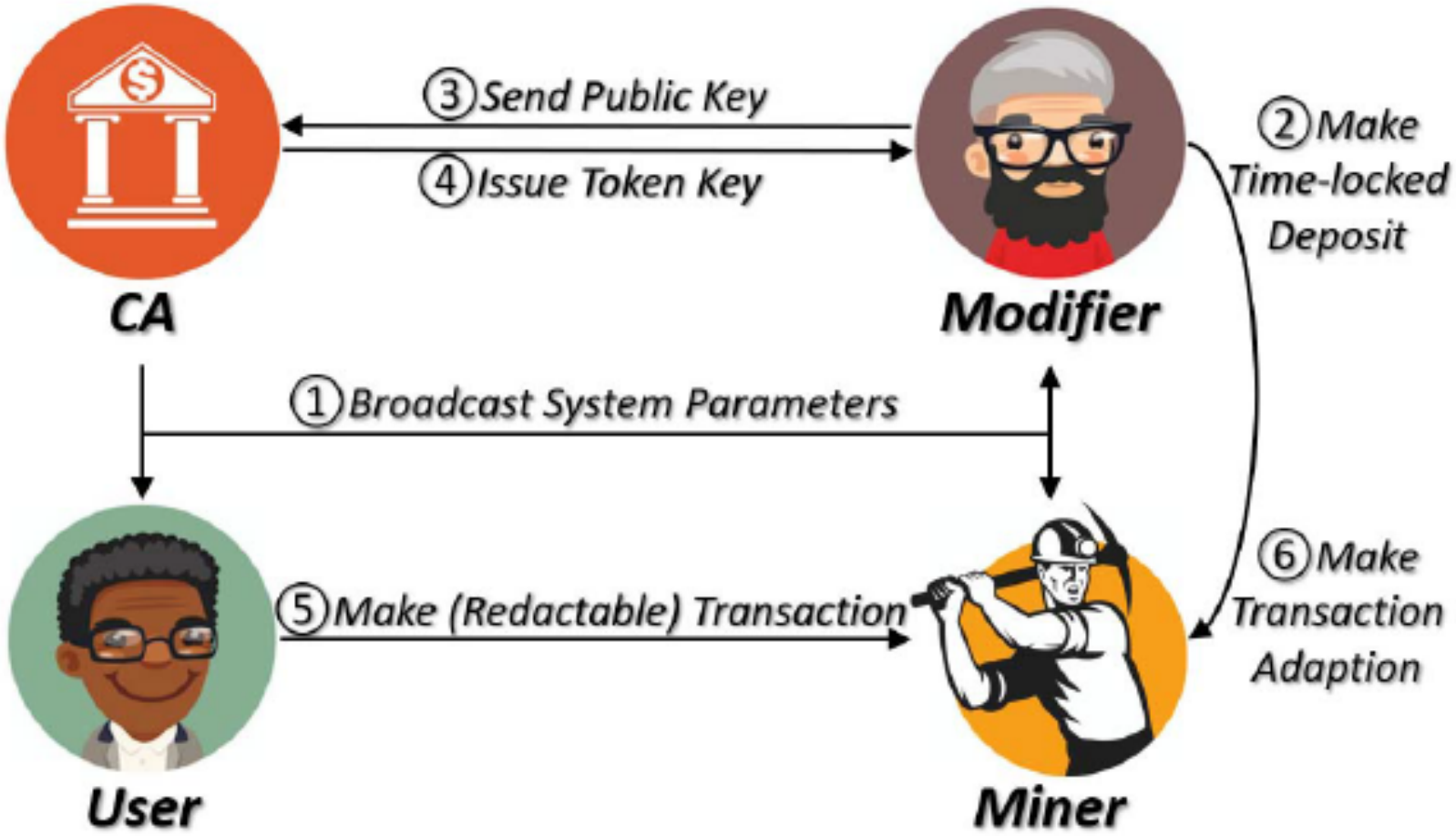


Fig. 1. System model.

KEBR *System Initialization*

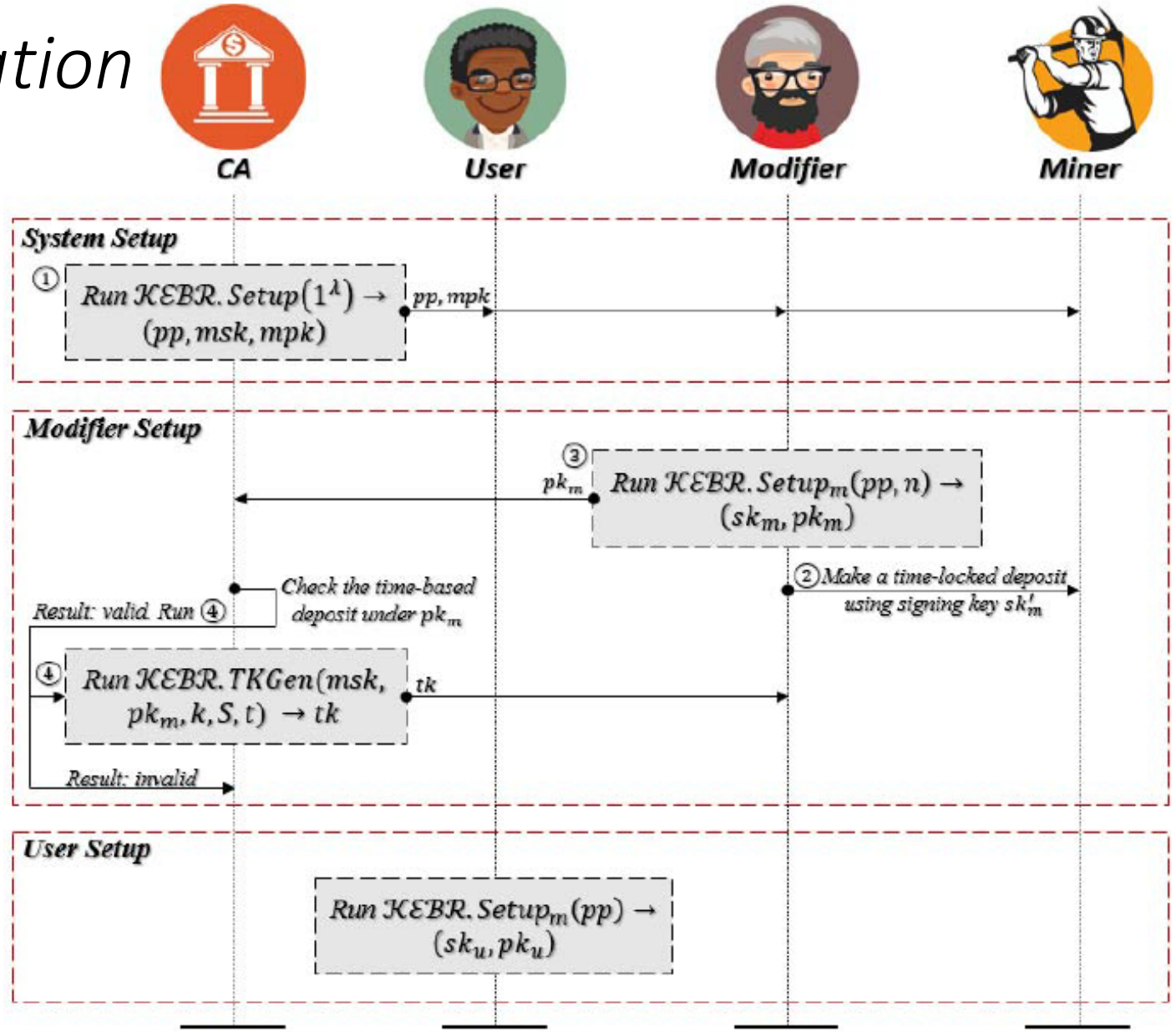


Fig. 2. System initialization in KEBR.

KEBR *Transaction Making*

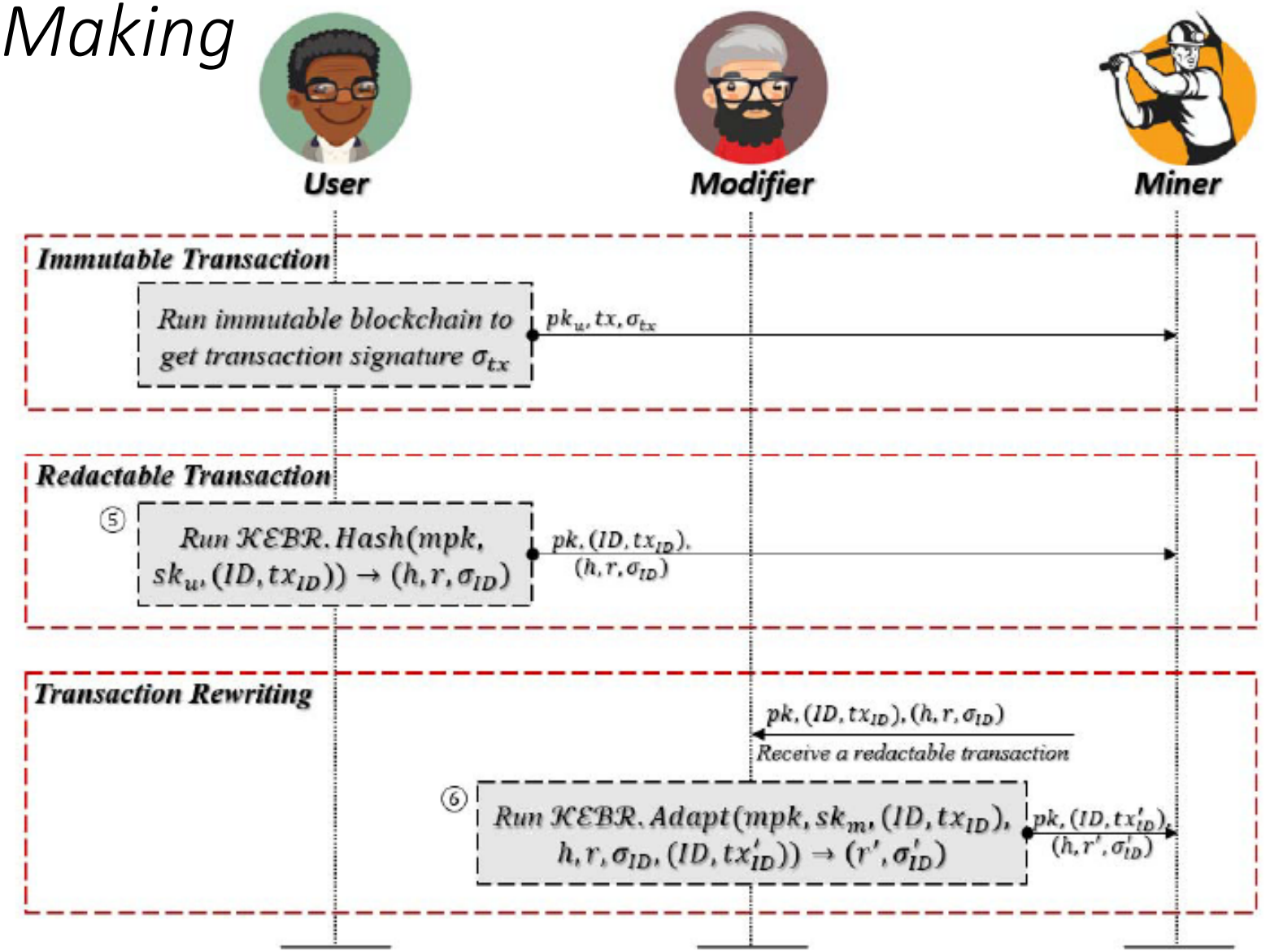


Fig. 3. Transaction making in KEBR.

KEBR *Transaction Verification*

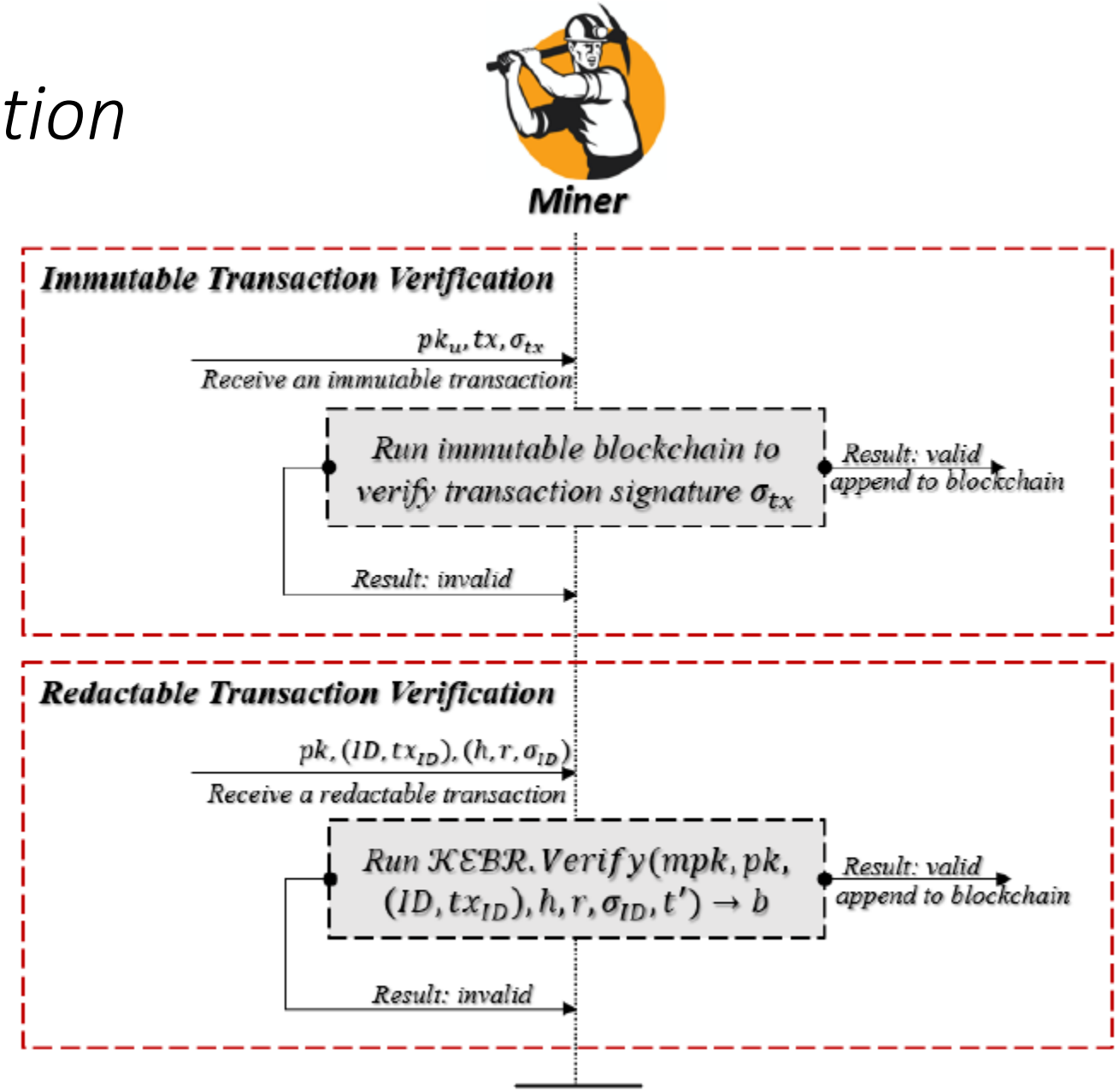


Fig. 4. Transaction verification in KEBR.

KEBR *Malicious Punishment*

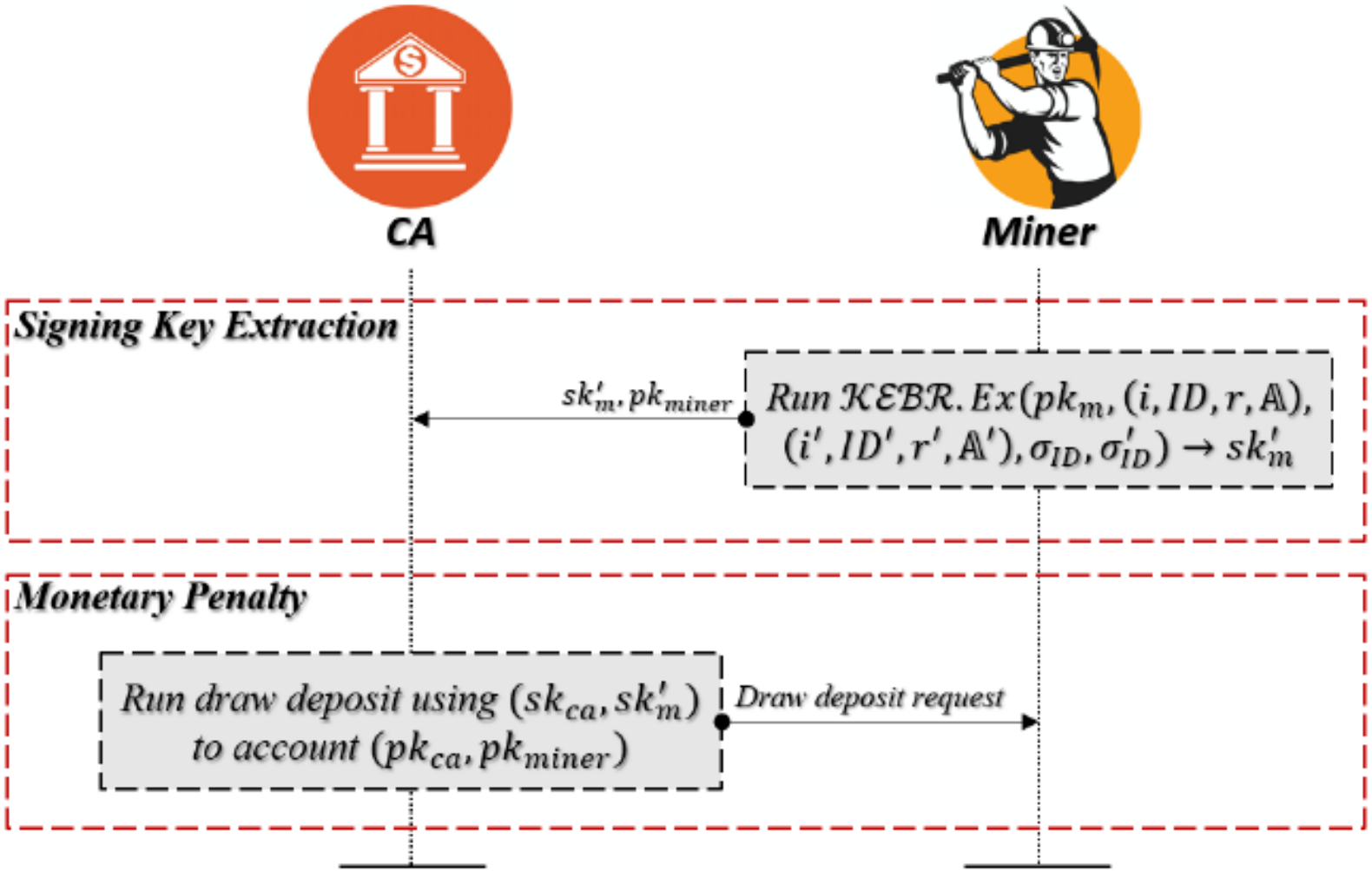


Fig. 5. Malicious punishment in KEBR.

Threat Model

- Assumption
  - CA is fully trusted.
  - Miners are majority trusted as the normal blockchain ecosystem.
  - Users and modifiers are untrusted and they can get together to launch collusion attacks.

Security Model

*Definition 9 (*EUF-CMA *Security): Let* $\mathcal{O}$ *denote a set of oracles: a modifier setup oracle* $\mathcal{O}_{\mathsf{Setup_m}}(\cdot,\cdot)$*, a modifier corrupt oracle* $\mathcal{O}_{\mathsf{Corrupt_m}}(\cdot)$*, a user setup oracle* $\mathcal{O}_{\mathsf{Setup_u}}(\cdot)$*, a user corrupt oracle* $\mathcal{O}_{\mathsf{Corrupt_u}}(\cdot)$*, a token key generation oracle* $\mathcal{O}_{\mathsf{TKGen}}(\cdot,\cdot,\cdot,\cdot)$*, a hash oracle* $\mathcal{O}_{\mathsf{Hash}}(\cdot,\cdot,\cdot)$*, and an adaption oracle* $\mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$*. The* EUF-CMA *security definition of a KERB is based on the experiment in Fig. 6.*

Security Model

$\mathbf{Exp}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda)$
$(pp, msk, mpk) \leftarrow \mathcal{KERB}.\mathsf{Setup}(1^\lambda)$;
$\mathcal{L}_{\mathsf{s_m}}, \mathcal{L}_{\mathsf{corr_m}}, \mathcal{L}_{\mathsf{s_u}}, \mathcal{L}_{\mathsf{corr_u}}, \mathcal{L}_{\mathsf{tk}}, \mathcal{L}_{\mathsf{h}}, \mathcal{L}_{\mathsf{apt}} \leftarrow \emptyset$;
$(pk^*, (\mathsf{ID}^*, tx_{\mathsf{ID}^*}), \mathbb{A}^*, h^*, r^*, \sigma_{\mathsf{ID}^*}, t^*) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, mpk)$;
return 1 if $\mathcal{KERB}.\mathsf{Verify}(mpk, pk^*, (\mathsf{ID}^*, tx_{\mathsf{ID}^*}), h^*, r^*, \sigma_{\mathsf{ID}^*}, t^*) = 1 \wedge$
$((pk^* = pk^*_u \wedge pk^*_u \notin \mathcal{L}_{\mathsf{corr_u}} \wedge (pk^*_u, (\mathsf{ID}^*, tx_{\mathsf{ID}^*}), \mathbb{A}^*, h^*, r^*, \sigma_{\mathsf{ID}^*}) \notin \mathcal{L}_{\mathsf{h}}) \vee$
$(pk^* = (pk^*_u, pk^*_m, tk^*) \wedge (pk^*_m \notin \mathcal{L}_{\mathsf{corr_m}} \vee tk^* \notin \mathcal{L}_{\mathsf{tk}}) \wedge$
$(pk^*_m, \cdot, h^*, \cdot, \cdot, (\mathsf{ID}^*, tx_{\mathsf{ID}^*}), r^*, \sigma_{\mathsf{ID}^*}) \notin \mathcal{L}_{\mathsf{apt}}))$,
else return 0.

Oracle $\mathcal{O}_{\mathsf{Setup_m}}(i, n)$
$(sk_m, pk_m) \leftarrow \mathcal{KERB}.\mathsf{Setup_m}(pp, n)$;
$\mathcal{L}_{\mathsf{s_u}} \leftarrow \mathcal{L}_{\mathsf{s_m}} \cup \{(i, sk_m, pk_m)\}$;
return pk_m.

Oracle $\mathcal{O}_{\mathsf{Setup_u}}(i)$
$(sk_u, pk_u) \leftarrow \mathcal{KERB}.\mathsf{Setup_u}(pp)$;
$\mathcal{L}_{\mathsf{s_u}} \leftarrow \mathcal{L}_{\mathsf{s_u}} \cup \{(i, sk_u, pk_u)\}$;
return pk_u.

Oracle $\mathcal{O}_{\mathsf{Corrupt_m}}(pk_m)$
$\mathcal{L}_{\mathsf{corr_u}} \leftarrow \mathcal{L}_{\mathsf{corr_m}} \cup \{pk_m\}$;
return sk_m.

Oracle $\mathcal{O}_{\mathsf{Corrupt_u}}(pk_u)$
$\mathcal{L}_{\mathsf{corr_u}} \leftarrow \mathcal{L}_{\mathsf{corr_u}} \cup \{pk_u\}$;
return sk_u.

Oracle $\mathcal{O}_{\mathsf{TKGen}}(pk_m, k, \mathcal{S}, t)$
$tk \leftarrow \mathcal{KERB}.\mathsf{TKGen}(msk, pk_m, k, \mathcal{S}, t)$;
$\mathcal{L}_{\mathsf{tk}} \leftarrow \mathcal{L}_{\mathsf{tk}} \cup \{(pk_m, k, \mathcal{S}, t)\}$;
return tk.

Oracle $\mathcal{O}_{\mathsf{Hash}}(pk_u, (\mathsf{ID}, tx_{\mathsf{ID}}), \mathbb{A})$
$(h, r, \sigma_{\mathsf{ID}}) \leftarrow \mathcal{KERB}.\mathsf{Hash}(mpk, sk_u, (\mathsf{ID}, tx_{\mathsf{ID}}), \mathbb{A})$;
$\mathcal{L}_{\mathsf{h}} \leftarrow \mathcal{L}_{\mathsf{h}} \cup \{(pk_u, (\mathsf{ID}, tx_{\mathsf{ID}}), \mathbb{A}, h, r, \sigma_{\mathsf{ID}})\}$;
return $(h, r, \sigma_{\mathsf{ID}})$.

Oracle $\mathcal{O}_{\mathsf{Adapt}}(pk_m, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, (\mathsf{ID}, tx'_{\mathsf{ID}}))$
$(r', \sigma'_{\mathsf{ID}}) \leftarrow \mathcal{KERB}.\mathsf{Adapt}(mpk, sk_m, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, (\mathsf{ID}, tx'_{\mathsf{ID}}))$;
$\mathcal{L}_{\mathsf{apt}} \leftarrow \mathcal{L}_{\mathsf{apt}} \cup \{(pk_m, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, (\mathsf{ID}, tx'_{\mathsf{ID}}), r', \sigma'_{\mathsf{ID}})\}$;
return $(r', \sigma'_{\mathsf{ID}})$.

Fig. 6. Experiment of EUF-CMA security in $\mathcal{KERB}$.

Security Model

We say that a KERB is EUF-CMA *secure, if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following advantage is negligible:* $\mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = \Pr[\mathbf{Exp}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = 1]$.

An interesting property of the KERB is k-time modification. It requires that whenever one obtains the messages about colliding redaction, as shown in Definition IV-A, one should be able to extract the secret signing key using the extraction algorithm. Hence, we give the security model called weak key extraction (w-KE security). The adversary wins the game if the extraction algorithm cannot extract the secret signing key from the colliding messages. Compared to the EUF-CMA security model, the w-KE security model allows $\mathcal{A}$ to corrupt all users and modifiers. Therefore, several oracles including modifier setup, modifier corrupt, user setup, and user corrupt oracles are not requested in the w-KE security model.

Security Model

*Definition 10 (*w-KE *Security): The* w-KE *security definition of a KERB is based on the experiment in Fig. 7. We say that a KERB is* w-KE *security, if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following advantage is negligible:* $\mathbf{Adv}^{\text{w-KE}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = \Pr[\mathbf{Exp}^{\text{w-KE}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = 1]$.

Our w-KE security model only extracts the secret signing key rather than the whole secret key since monetary penalty only requires the secret signing key in the blockchain setting. Specifically, the leakage of the secret signing key represents the loss of the time-locked deposit. The strong security notation for extracting the whole secret key is an interesting work, but it may influence performance. Hence, w-KE security is enough for the efficient KERB system.

Security Model

$\mathbf{Exp}^{\text{w-KE}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda)$

$\mathcal{L}_{\mathsf{tk}}, \mathcal{L}_{\mathsf{h}} \leftarrow \emptyset;$

$(pp, msk, mpk) \leftarrow \mathcal{KERB}.\mathsf{Setup}(1^\lambda);$

$\{pk_i^*, (\mathsf{ID}_i^*, tx_{\mathsf{ID}_i^*}), h_i^*, r_i^*, \mathbb{A}_i^*, \sigma_{\mathsf{ID}_i^*}, t_i^*\}_{i\in[k+1]} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{TKGen}}(\cdot,\cdot,\cdot,\cdot),\mathcal{O}_{\mathsf{H}}(\cdot)}(pp, mpk);$

return 1 if $(pk_m^*, k, \cdot, t^*) \in \mathcal{L}_{\mathsf{tk}} \ \wedge \ (\forall i \in [k+1] : pk_i^* = (\cdot, pk_m^*, tk^*) \ \wedge$

$\mathcal{KERB}.\mathsf{Verify}(mpk, pk_i^*, (\mathsf{ID}_i^*, tx_{\mathsf{ID}_i^*}), h_i^*, r_i^*, \sigma_{\mathsf{ID}_i^*}, t_i^*) = 1) \ \wedge$

$\forall i \in [k+1], \forall j \in [k+1], i \neq j :$

$\mathcal{KERB}.\mathsf{Ex}(pk_m^*, (i_i^*, \mathsf{ID}_i^*, r_i^*, \mathbb{A}_i^*), (i_j^*, \mathsf{ID}_j^*, r_j^*, \mathbb{A}_j^*), \sigma_{\mathsf{ID}_i^*}, \sigma_{\mathsf{ID}_j^*}) = \perp,$

else return 0.

Oracle $\mathcal{O}_{\mathsf{TKGen}}(pk_m, k, \mathcal{S}, t)$

$tk \leftarrow \mathcal{KERB}.\mathsf{TKGen}(msk, pk_m, k, \mathcal{S}, t);$

$\mathcal{L}_{\mathsf{tk}} \leftarrow \mathcal{L}_{\mathsf{tk}} \cup \{(pk_m, k, \mathcal{S}, t)\};$

return tk.

Oracle $\mathcal{O}_{\mathsf{H}}(m)$

$h \leftarrow H(m);$

$\mathcal{L}_{\mathsf{h}} \leftarrow \mathcal{L}_{\mathsf{h}} \cup \{m\};$

return h.

Fig. 7. Experiment of w-KE security in $\mathcal{KERB}$.

Proposed Scheme

Let $\mathcal{DS}$ = {Setup, KeyGen, Sign, Verify} be an EUF-CMA secure digital signature, and $\mathcal{CH}$ = {Setup, KeyGen, Hash, Adapt} be any chameleon hash. Following is the construction of $\mathcal{KERB}$:

$\mathcal{KERB}$.Setup$(1^\lambda) \rightarrow (pp, msk, mpk)$: The setup algorithm initializes public parameters of a digital signature $pp_{\mathcal{DS}} \leftarrow \mathcal{DS}$.Setup$(1^\lambda)$, a chameleon hash $pp_{\mathcal{CH}} \leftarrow \mathcal{CH}$.Setup$(1^\lambda)$, a key-pair $(sk_a, pk_a) \leftarrow \mathcal{DS}$.KeyGen$(pp_{\mathcal{DS}})$ for a digital signature and a key-pair $(sk_h, pk_h) \leftarrow \mathcal{CH}$.KeyGen$(pp_{\mathcal{CH}})$ for a chameleon hash. Then, it picks a collision-resistant hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The algorithm returns a public parameter $pp = (pp_{\mathcal{DS}}, pp_{\mathcal{CH}})$, a master secret key $msk = sk_a$ and a master public key $mpk = (sk_h, pk_a, pk_h, H)$.

Proposed Scheme

$\mathcal{KERB}.\mathsf{Setup_m}(pp, n) \rightarrow (sk_m, pk_m)$: The modifier setup algorithm initializes a signature key-pair $(sk'_m, pk'_m) \leftarrow \mathcal{DS}.\mathsf{KeyGen}(pp_{\mathcal{DS}})$. It then randomly chooses terms $\alpha, \rho_1, \rho_2, \ldots, \rho_n, r_1, r_2, \ldots, r_n \in \mathbb{Z}_p$, computes $c_0 = g^{\alpha}$ and for each $i \in [n] : c_i = (c_{1,i}, c_{2,i}) = (g^{r_i}, c_0^{r_i} g^{\rho_i})$. The algorithm returns a secret key $sk_m = (sk'_m, \{r_i, \rho_i\}_{i \in [n]})$ and a public key $pk_m = (pk'_m, c_0, \{c_i\}_{i \in [n]})$.

$\mathcal{KERB}.\mathsf{Setup_u}(pp) \rightarrow (sk_u, pk_u)$: The user setup algorithm initializes a signature key-pair $(sk_u, pk_u) \leftarrow \mathcal{DS}.\mathsf{KeyGen}(pp_{\mathcal{DS}})$. The algorithm returns a secret key sk_u and a public key pk_u.

Proposed Scheme

$\mathcal{KERB}.\mathsf{TKGen}(msk, pk_m, k, \mathcal{S}, t) \rightarrow tk$: Parse $pk_m = (pk'_m, c_0, \{c_i\}_{i\in[n]})$, where n is the size of c_i. The token key generation algorithm returns a failure symbol $\perp$ if no time-locked deposit that can only be spent before time t if valid signatures signed by sk_a and sk'_m; otherwise, it sets $pk_m = (pk'_m, c_0, \{c_i\}_{i\in[k]})$ if $n \geq k$. The algorithm returns a token key $tk \leftarrow \mathcal{DS}.\mathsf{Sign}(sk_a, (pk_m, \mathcal{S}, t))$.

$\mathcal{KERB}.\mathsf{Hash}(mpk, sk_u, (\mathsf{ID}, tx_{\mathsf{ID}}), \mathbb{A}) \rightarrow (h, r, \sigma_{\mathsf{ID}})$: The hash algorithm picks a randomness r in the chameleon hash randomness domain and generates a hash value $h \leftarrow \mathcal{CH}.\mathsf{Hash}(pk_h, (\mathsf{ID}, tx_{\mathsf{ID}}); r)$. It then defines the access policy $\mathbb{A}$ of redaction by generating a signature $\sigma_{\mathsf{ID}} \leftarrow \mathcal{DS}.\mathsf{Sign}(sk_u, (\mathsf{ID}, r, \mathbb{A}))$. Note that

- h and σ_{ID} are linked via the transaction identity ID, and
- σ_{ID} excludes the transaction information tx_{ID} to prevent anyone to infer the previous transaction from the redacted transaction.

Proposed Scheme

The algorithm returns a hash value h, a randomness r and a signature σ_{ID}.

$\mathcal{KERB}.\mathsf{Adapt}(mpk, sk_m, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, (\mathsf{ID}, tx'_{\mathsf{ID}})) \rightarrow (r', \sigma'_{\mathsf{ID}})$: The adaption algorithm generates a randomness $r' \leftarrow \mathcal{CH}.\mathsf{Adapt}(sk_h, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, (\mathsf{ID}, tx'_{\mathsf{ID}}))$ and picks an index i that is never used before to generate a signature $\sigma' \leftarrow \mathcal{DS}.\mathsf{Sign}(sk'_m, (\mathsf{ID}, r', \mathbb{A}))$. To convince no more than k-time redaction, it computes $z = \rho_i \cdot H(i, \mathsf{ID}, r', \mathbb{A}) + sk'_m$, $c'_{2,i} = c_{2,i} \cdot (pk_m \cdot g^{-z})^{1/H(i,\mathsf{ID},r',\mathbb{A})}$, randomly picks $v \in \mathbb{Z}_p$ and calculates $s = v - r_i c$ and $c = H(g, c_0, g^v, c_0^v, pk_m, c_{1,i}, c'_{2,i}, \sigma')$. The algorithm returns a randomness r' and a signature $\sigma'_{\mathsf{ID}} = (\sigma', z, \pi)$, where $\pi = (i, s, g^v, c_0^v)$.

Proposed Scheme

$\mathcal{KERB}.\mathsf{Verify}(mpk, pk, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r, \sigma_{\mathsf{ID}}, t') \rightarrow b$: Let r' be the part of messages signed in the signature σ_{ID}. If r and r' are equal, the transaction tx_{ID} has not been redacted and the input public key pk comes from the user pk_u, the verifier returns 0 if anyone of the following conditions is not true:

- $\mathcal{CH}.\mathsf{Verify}(pk_h, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r) = 1$;
- $\mathcal{DS}.\mathsf{Verify}(pk_u, \sigma_{\mathsf{ID}}, (\mathsf{ID}, r, \mathbb{A})) = 1$.

If r and r' are not equal, the transaction tx_{ID} has been redacted and the input public key of the user and modifier pk including the public keys (pk_u, pk_m) and the token key tk of the modifier, the verifier returns 0 if anyone of the following conditions is not true:

- $\mathcal{CH}.\mathsf{Verify}(pk_h, (\mathsf{ID}, tx_{\mathsf{ID}}), h, r) = 1$;
- $\mathcal{DS}.\mathsf{Verify}(pk_u, \sigma_{\mathsf{ID}}, (\mathsf{ID}, r, \mathbb{A})) = 1$;
- $\mathcal{DS}.\mathsf{Verify}(pk'_m, \sigma', (i, \mathsf{ID}, r', \mathbb{A})) = 1$;
- $\mathcal{DS}.\mathsf{Verify}(pk_a, tk, (pk_m, \mathcal{S}, t)) = 1$;
- $\mathcal{S} \models \mathbb{A}$ and $t' < t$ & $g^v = g^s c_{1,i}^c$ and $c_0^v = c_0^s c_{2,i}^{\prime c}$.

The algorithm returns 1 if there is no any failure during the above verifications; otherwise, returns 0.

$\mathcal{KERB}.\mathsf{Ex}(pk_m, (i, \mathsf{ID}, r, \mathbb{A}), (i', \mathsf{ID}', r', \mathbb{A}'), \sigma_{\mathsf{ID}}, \sigma_{\mathsf{ID}'}) \rightarrow sk'_m$: The extraction algorithm returns a failure symbol $\perp$ if anyone of the following conditions is false:

- $i = i'$ and $(\mathsf{ID}, r, \mathbb{A}) \neq (\mathsf{ID}', r', \mathbb{A}')$;
- $\mathcal{DS}.\mathsf{Verify}(pk'_m, \sigma, (i, \mathsf{ID}, r, \mathbb{A})) = 1$;
- $\mathcal{DS}.\mathsf{Verify}(pk'_m, \sigma', (i', \mathsf{ID}', r', \mathbb{A}')) = 1$.

The algorithm returns $sk'_m = \frac{z \cdot H(i', \mathsf{ID}', r', \mathbb{A}') - z' \cdot H(i, \mathsf{ID}, r, \mathbb{A})}{H(i', \mathsf{ID}', r', \mathbb{A}') \cdot H(i, \mathsf{ID}, r, \mathbb{A})}$.

Proposed Scheme

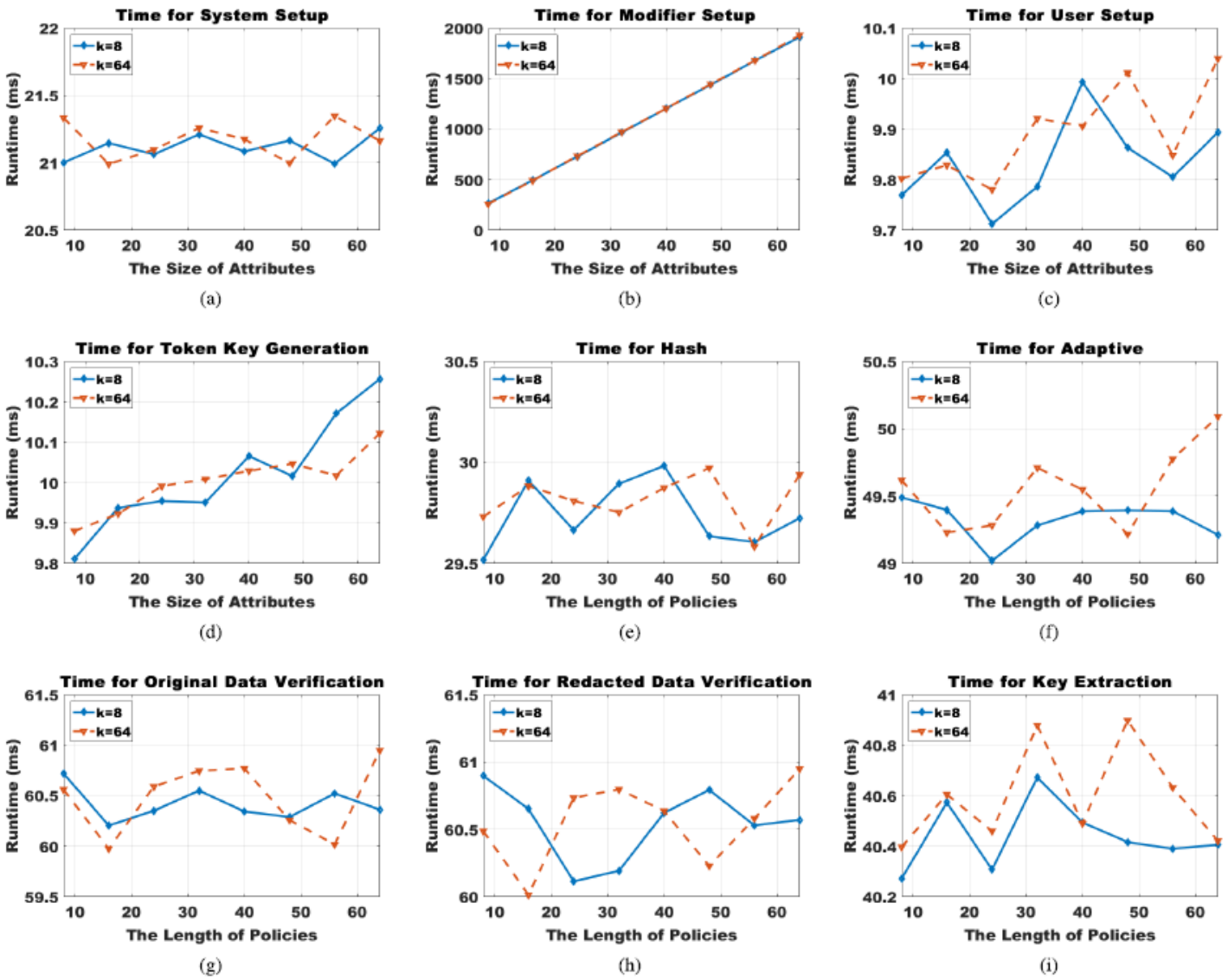


Fig. 8. Experimental performances.

Security Proof

Theorem 1: Suppose the underlying digital signature scheme $\mathcal{DS}$ is EUF-CMA *secure, then the proposed semi-generic construction $\mathcal{KERB}$ is* EUF-CMA *secure with the following advantage:* $\mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = \mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{A},\mathcal{DS}}(1^\lambda)$.

Theorem 2: Suppose the soundness of DDH assumption holds and the underlying hash function is collision-resistant, then the proposed semi-generic construction $\mathcal{KERB}$ is w-KE *secure with the following advantage:* $\mathbf{Adv}^{\text{w-KE}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = \epsilon_{\mathsf{s}_{\mathsf{DDH}}}$ *and* $\mathbf{Adv}^{\text{w-KE}}_{\mathcal{A},\mathcal{KERB}}(1^\lambda) = \epsilon_{\mathsf{h}}$, *where $\epsilon_{\mathsf{s}_{\mathsf{DDH}}}$ and ϵ_{h} are the advantages of breaking the soundness of DDH and the collision-resistance of hash functions.*

Security Proof

APPENDIX A
CORRECTNESS OF OUR PROPOSED SCHEME

In the adaption algorithm, we apply the Schnorr signature to realize the k-time data redaction. Specifically, we have $g^v = g^s c_{1,i}^c$ and $c_0^v = c_0^s c_{2,i}^{rc}$ as:

$$\begin{aligned} g^s c_{1,i}^c &= g^{v-r_i c}(g^{r_i})^c = g^v, \\ c_0^s c_{2,i}^{rc} &= c_0^{v-r_i c}(c_0^{r_i} g^{\rho_i} \cdot (pk_m \cdot g^{-z})^{1/H(i,\mathsf{ID},r',\mathbb{A})})^c \\ &= c_0^{v-r_i c}(c_0^{r_i})^c = c_0^v. \end{aligned}$$

If the modifier redacts the information more than k-time, the secret signing key can be extracted. Let $h = H(i, \mathsf{ID}, r, \mathbb{A})$ and $h' = H(i', \mathsf{ID}', r', \mathbb{A}')$, we have $\rho_i = \rho_i'$ that can extract the secret signing key sk_m' as:

$$\begin{aligned} &\frac{z \cdot H(i', \mathsf{ID}', r', \mathbb{A}') - z' \cdot H(i, \mathsf{ID}, r, \mathbb{A})}{H(i', \mathsf{ID}', r', \mathbb{A}') \cdot H(i, \mathsf{ID}, r, \mathbb{A})} \\ &= \frac{z \cdot h' - z' \cdot h}{h'h} = \frac{(\rho_i \cdot h + sk_m') \cdot h' - (\rho_i' \cdot h' + sk_m') \cdot h}{h'h} \\ &= \frac{\rho_i hh' + sk_m' h' - \rho_i' hh' - sk_m' h}{h'h} = \frac{sk_m'(h'h)}{h'h} = sk_m'. \end{aligned}$$