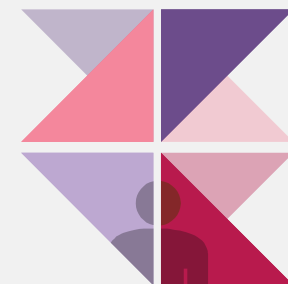


Practical UC-Secure Delegatable Credentials with Attributes and Their Application to Blockchain

实用的 UC 安全委托证书以及其属性和在区块链中的应用

2017 CCS





目录

content

01. 介绍

Introduction

04. 许可区块链上的应用

Application to permissioned blockchains

02. 委托证书通用框架

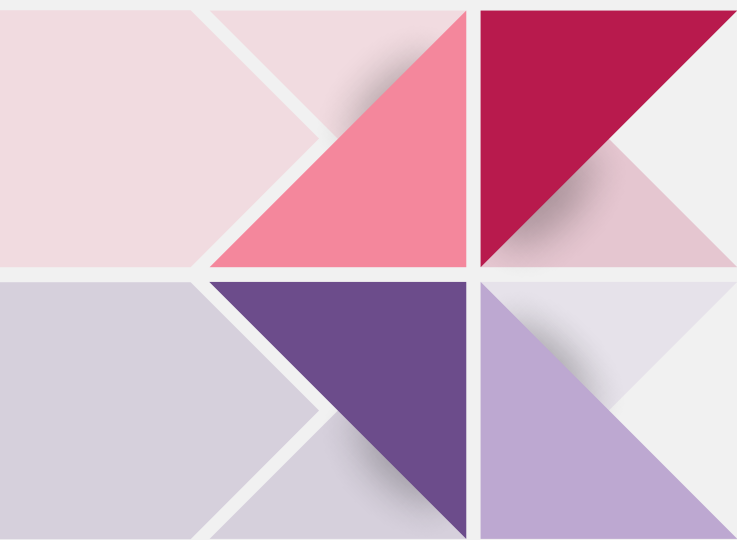
A generic construction for
delegatable credentials

05. 论文总结

Conclusion

03. 使用配对的具体实例

A concrete instantiation using pairings



01

介绍

Introduction

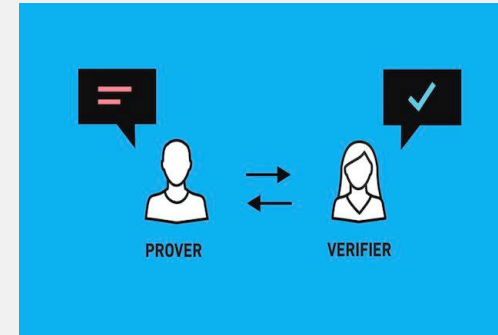
01 介绍

Introduction

UC-Secure


Zero Knowledge Proof


Token



Drivers Licenses





 Local authority

 Regional authority

(permissioned) blockchain



 Organizations(CAs)

 parties that are allowed to submit transactions

01 介绍

Introduction

DAC (Delegatable anonymous credentials)

A few DAC constructions have been proposed, but none is suitable for practical use for the following reasons:

- 1、 Generic zero-knowledge proofs or Groth-Sahai proofs with many expensive pairing operations and a large credential size.
- 2、 Blackbox fashion
- 3、 Do not consider attributes
- 4、 Either do not provide an deal functionality for DAC or are proven secure in standalone models that guarantee security only if a protocol is run in isolation

01 介绍

Introduction

Contribution Summary

- 1、 we first provide a (surprisingly simple) ideal functionality \mathcal{F}_{dac} for delegatable credentials with attributes.
- 2、 we propose a generic DAC construction from signature schemes and zero-knowledge proofs and prove it secure in the universal composability (UC) framework
- 3、 we describe a very efficient instantiation of our DAC scheme
- 4、 we report on an implementation of our scheme in the context of a privacy-preserving membership service for permissioned blockchains

01 介绍

Introduction

UC-Secure



Universal Composability (UC) framework

- 利用统一的分析框架解决密码学任务的组合执行问题
- Real World Model & Ideal World Model

Real World Model	Ideal World Model
π 表示需要被分析的协议，被抽象为交互式图灵机；	理想功能F
Z表示所分析协议所处的运行环境，也被抽象为一个实体，用于模型化除被分析的协议之外的所有其他协议（例如，调用当前协议的上层协议等）；	Z表示所分析协议所处的运行环境； 理想功能F是一个特殊的实体，刻画了密码任务应该实现的功能和最大允许的信息泄露，并且通过安全信道直接与（虚拟的）协议参与方进行交互；
A表示概率多项式时间（Probabilistic Polynomial Time, PPT）攻击者，控制着所有协议用户之间的通信网络。	理想攻击者S不能直接控制协议用户之间的通信，仅仅能通过和F交互来获取一些不会影响最终安全目标的信息，从形式上确保了理想模型中密码任务的安全性。
	Π 是调用了理想功能F的任何（混合）密码协议

01 介绍

Introduction

Ideal Functionality \mathcal{F}_{crs} .

Functionality \mathcal{F}_{crs}

- (1) When receiving input (CRS, sid) from party \mathcal{P} , first verify that $\text{sid} = (\{\mathcal{P}\}, \text{sid}')$ where $\{\mathcal{P}\}$ is a set of identities, and that $\mathcal{P} \in \{\mathcal{P}\}$; else ignore the input. Next, if there is no value r recorded then choose and record $r \xleftarrow{\$} \mathcal{D}$. Finally, send a public delayed output $(\text{CRS}, \text{sid}, r)$ to \mathcal{P} .

Ideal Functionality \mathcal{F}_{ca} .

Functionality \mathcal{F}_{ca}

- (1) Upon receiving the first message $(\text{REGISTER}, \text{sid}, v)$ from party \mathcal{P} , send $(\text{REGISTERED}, \text{sid}, v)$ to the adversary \mathcal{A} ; upon receiving OK from \mathcal{A} , and if $\text{sid} = \mathcal{P}$ and this is the first request from \mathcal{P} , then record the pair (\mathcal{P}, v) .
- (2) Upon receiving a message $(\text{RETRIEVE}, \text{sid})$ from party \mathcal{P}' , send $(\text{RETRIEVE}, \text{sid}, \mathcal{P}')$ to \mathcal{A} , and wait for an OK from \mathcal{A} . Then, if there is a recorded pair (sid, v) output $(\text{RETRIEVE}, \text{sid}, v)$ to \mathcal{P}' . Else output $(\text{RETRIEVE}, \text{sid}, \perp)$ to \mathcal{P}' .

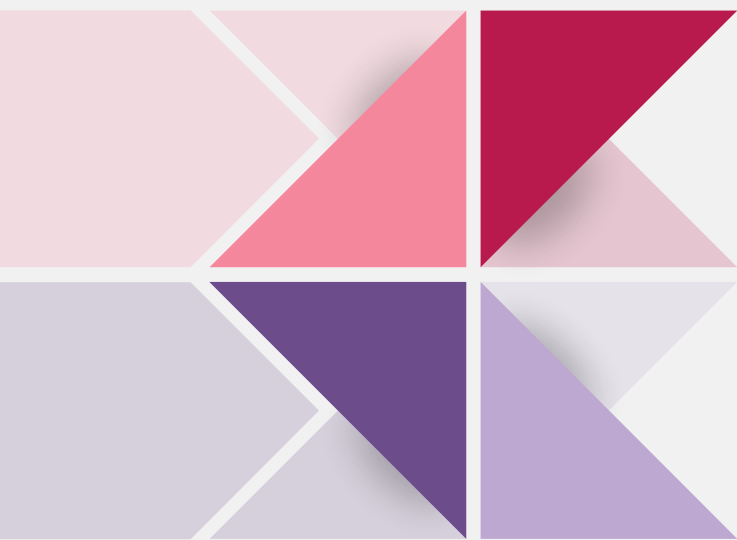
01 介绍

Introduction

Ideal Functionality \mathcal{F}_{smt}

Functionality \mathcal{F}_{smt}

- (1) On input (SEND, sid , m) from a party \mathcal{P} , abort if $sid \neq (\mathcal{S}, \mathcal{R}, sid')$, send (SEND, sid , $l(m)$) to the adversary, generate a private delayed output (SENT, sid , m) to \mathcal{R} and halt.



02

委托证书通用框架

A generic construction for delegatable credentials

02 委托证书通用框架

A generic construction for delegatable credentials

委托证书的定义

- (1) Setup. On input (SETUP, sid , $\langle n_i \rangle_i$) from \mathcal{I} .
 - Verify that $sid = (\mathcal{I}, sid')$.
 - Output (SETUP, sid , $\langle n_i \rangle_i$) to \mathcal{A} and wait for response (SETUP, sid , Present, Ver, $\langle \mathbb{A}_i \rangle_i$) from \mathcal{A} .
 - Store algorithms Present and Ver and credential parameters $\langle \mathbb{A}_i \rangle_i$, $\langle n_i \rangle_i$, initialize $\mathcal{L}_{de} \leftarrow \emptyset$; $\mathcal{L}_{at} \leftarrow \emptyset$.
 - Output (SETUPDONE, sid) to \mathcal{I} .
- (2) Delegate. On input (DELEGATE, sid , $ssid$, $\vec{a}_1, \dots, \vec{a}_L$, \mathcal{P}_j) from some party \mathcal{P}_i , with $\vec{a}_L \in \mathbb{A}_L^{n_L}$.
 - If $L = 1$, check $sid = (\mathcal{P}_i, sid')$ and add an entry $\langle \mathcal{P}_j, \vec{a}_1 \rangle$ to \mathcal{L}_{de} .
 - If $L > 1$, check that an entry $\langle \mathcal{P}_i, \vec{a}_1, \dots, \vec{a}_{L-1} \rangle$ exists in \mathcal{L}_{de} .
 - Output (ALLOWDEL, sid , $ssid$, \mathcal{P}_i , \mathcal{P}_j , L) to \mathcal{A} and wait for input (ALLOWDEL, sid , $ssid$) from \mathcal{A} .
 - Add an entry $\langle \mathcal{P}_j, \vec{a}_1, \dots, \vec{a}_L \rangle$ to \mathcal{L}_{de} .
 - Output (DELEGATE, sid , $ssid$, $\vec{a}_1, \dots, \vec{a}_L$, \mathcal{P}_i) to \mathcal{P}_j .
- (3) Present. On input (PRESENT, sid , m , $\vec{a}_1, \dots, \vec{a}_L$) from some party \mathcal{P}_i , with $\vec{a}_i \in (\mathbb{A}_i \cup \perp)^{n_i}$ for $i = 1, \dots, L$.
 - Check that an entry $\langle \mathcal{P}_i, \vec{a}'_1, \dots, \vec{a}'_L \rangle$ exists in \mathcal{L}_{de} such that $\vec{a}_i \leq \vec{a}'_i$ for $i = 1, \dots, L$.
 - Set $at \leftarrow \text{Present}(m, \vec{a}_1, \dots, \vec{a}_L)$ and abort if $\text{Ver}(at, m, \vec{a}_1, \dots, \vec{a}_L) = 0$.
 - Store $\langle m, \vec{a}_1, \dots, \vec{a}_L \rangle$ in \mathcal{L}_{at} .
 - Output (TOKEN, sid , at) to \mathcal{P}_i .
- (4) Verify. On input (VERIFY, sid , at , m , $\vec{a}_1, \dots, \vec{a}_L$) from some party \mathcal{P}_i .
 - If there is no record $\langle m, \vec{a}_1, \dots, \vec{a}_L \rangle$ in \mathcal{L}_{at} , \mathcal{I} is honest, and for $i = 1, \dots, L$, there is no corrupt \mathcal{P}_j such that $\langle \mathcal{P}_j, \vec{a}'_1, \dots, \vec{a}'_i \rangle \in \mathcal{L}_{de}$ with $\vec{a}_j \leq \vec{a}'_j$ for $j = 1, \dots, i$, set $f \leftarrow 0$.
 - Else, set $f \leftarrow \text{Ver}(at, m, \vec{a}_1, \dots, \vec{a}_L)$.
 - Output (VERIFIED, sid , f) to \mathcal{P}_i .

Figure 1: Ideal functionality for delegatable credentials with attributes \mathcal{F}_{dac}

02 委托证书通用框架

A generic construction for delegatable credentials

结构概览

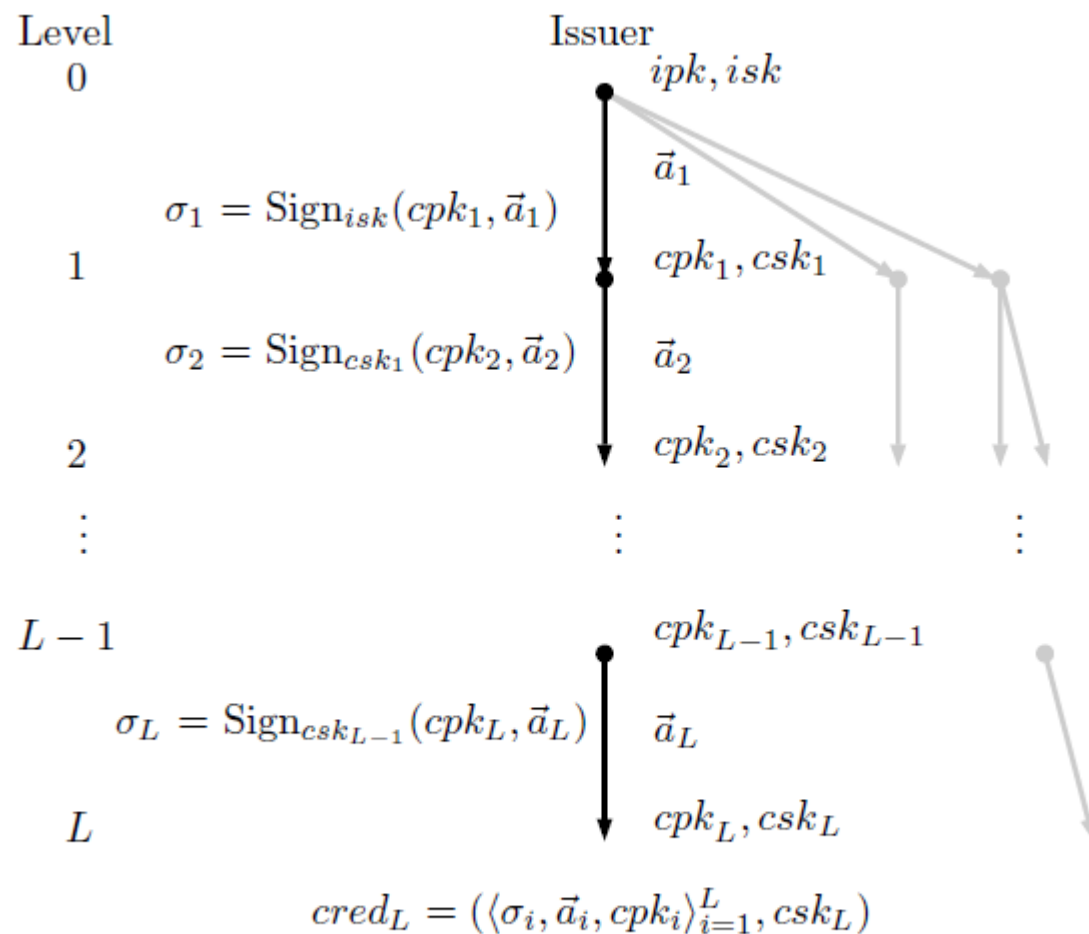


Figure 2: Our Generic Construction: Delegation

02 委托证书通用框架

A generic construction for delegatable credentials

Signature Schemes

PPT(Probabilistic Polynomial Time) algorithms

Sig =(Setup, Gen, Sign, Verify)

$\text{Sig.Setup}(1^\kappa) \xrightarrow{\$} sp$: The setup algorithm takes as input a security parameter and outputs public system parameters that also specify a message space \mathcal{M} .

$\text{Sig.Gen}(sp) \xrightarrow{\$} (sk, vk)$: The key generation algorithm takes as input system parameters and outputs a verification key vk and a corresponding secret key sk .

$\text{Sig.Sign}(sk, m) \xrightarrow{\$} \sigma$: The signing algorithm takes as input a private key sk and a message $m \in \mathcal{M}$ and outputs a signature σ .

$\text{Sig.Verify}(vk, m, \sigma) \rightarrow 1/0$: The verification algorithm takes as input a public verification key vk , a message m and a signature σ and outputs 1 for acceptance or 0 for rejection according to the input.

02 委托证书通用框架

A generic construction for delegatable credentials

Sibling Signatures

Sib = (Setup, Gen, Sign₁, Sign₂, Verify₁, Verify₂)

Sib.Setup(1^κ) $\xrightarrow{\$}$ sp : The setup algorithm takes as input a security parameter and outputs public system parameters that also specify two message spaces \mathcal{M}_1 and \mathcal{M}_2 .

Sib.Gen(sp) $\xrightarrow{\$}$ (sk, vk) : The key generation algorithm takes as input system parameters and outputs a verification key vk and a corresponding secret key sk .

Sib.Sign₁(sk, m) $\xrightarrow{\$}$ σ : The signing algorithm takes as input a private key sk and a message $m \in \mathcal{M}_1$ and outputs a signature σ .

Sib.Sign₂(sk, m) $\xrightarrow{\$}$ σ : The signing algorithm takes as input a private key sk and a message $m \in \mathcal{M}_2$ and outputs a signature σ .

Sib.Verify₁(vk, m, σ) \rightarrow 1/0 : The verification algorithm takes as input a public verification key vk , a message m and a signature σ and outputs 1 for acceptance or 0 for rejection according to the input.

Sib.Verify₂(vk, m, σ) \rightarrow 1/0 : The verification algorithm takes as input a public verification key vk , a message m and a signature σ and outputs 1 for acceptance or 0 for rejection according to the input.

02 委托证书通用框架

A generic construction for delegatable credentials

通用架构 Π_{dac}

Setup

(1) \mathcal{I} , upon receiving input (SETUP, sid , $\langle n_i \rangle_i$):

- Check that $sid = \mathcal{I}, sid'$ for some sid' .
- Run $(ipk, isk) \leftarrow \text{Sib}_0.\text{Gen}(1^\kappa)$ and compute proof $\pi_{isk} \leftarrow \text{NIZK}\{\boxed{isk} : (ipk, isk) \in \text{Sib}_0.\text{Gen}(1^\kappa)\}$. Register public key (ipk, π_{isk}) with \mathcal{F}_{ca} . Let $cpk_0 \leftarrow ipk$.
- Output (SETUPDONE, sid).

(2) \mathcal{P}_i on input (DELEGATE, sid , $ssid$, $\vec{a}_1, \dots, \vec{a}_L, \mathcal{P}_j$) with $\vec{a}_L \in \mathbb{A}_L^{n_L}$:

- If $L = 1$, \mathcal{P}_i only proceeds if he is the issuer \mathcal{I} with $sid = (\mathcal{I}, sid')$. If $L > 1$, \mathcal{P}_i checks that he possesses a credential chain that signs $\vec{a}_1, \dots, \vec{a}_{L-1}$. That is, he looks up a record $cred = (\langle \sigma_i, \vec{a}_i, cpk_i \rangle_{i=1}^{L-1}, csk_{L-1})$ in $\mathcal{L}_{\text{cred}}$.
- Send $(sid, ssid, \vec{a}_1, \dots, \vec{a}_L)$ to \mathcal{P}_j over \mathcal{F}_{smt} .
- \mathcal{P}_j , upon receiving $(sid, ssid, \vec{a}_1, \dots, \vec{a}_L)$ to \mathcal{P}_j over \mathcal{F}_{smt} from \mathcal{P}_i , generate a fresh credential specific key pair $(cpk_L, csk_L) \leftarrow \text{Sib}_L.\text{Gen}(1^\kappa)$.
- Send cpk_L to \mathcal{P}_i over \mathcal{F}_{smt} .
- \mathcal{P}_i , upon receiving cpk_L from \mathcal{P}_j over \mathcal{F}_{smt} , computes $\sigma_L \leftarrow \text{Sib}_{L-1}.\text{Sign}_1(csk_{L-1}; cpk_L, \vec{a}_L)$ and sends $\langle \sigma_i, cpk_i \rangle_{i=1}^L$ to \mathcal{P}_j over \mathcal{F}_{smt} .
- \mathcal{P}_j , upon receiving $\langle \sigma_i, cpk_i \rangle_{i=1}^L$ from \mathcal{P}_i over \mathcal{F}_{smt} , verifies $\text{Sib}_{i-1}.\text{Verify}_1(cpk_{i-1}, \sigma_i, cpk_i, \vec{a}_i)$ for $i = 1, \dots, L$. It stores $cred \leftarrow (\langle \sigma_i, \vec{a}_i, cpk_i \rangle_{i=1}^L, csk_L)$ in $\mathcal{L}_{\text{cred}}$. Output (DELEGATE, sid , $ssid$, $\vec{a}_1, \dots, \vec{a}_L, \mathcal{P}_i$).

Delegate

02 委托证书通用框架

A generic construction for delegatable credentials

通用架构

Present

(3) \mathcal{P}_i , upon receiving input (PRESENT, $sid, m, \vec{a}_1, \dots, \vec{a}_L$) with $\vec{a}_i \in (\mathbb{A}_i \cup \perp)^{n_i}$ for $i = 1, \dots, L$:

- Look up a credential $cred = (\langle \sigma_i, \vec{a}'_i, cpk_i \rangle_{i=1}^L, csk_L)$ in \mathcal{L}_{cred} , such that $\vec{a}_i \leq \vec{a}'_i$ for $i = 1, \dots, L$. Abort if no such credential was found.
- Create an attribute token by proving knowledge of the credential:

$at \leftarrow \text{NIZK}\{(\sigma_1, \dots, \sigma_L, cpk_1, \dots, cpk_L, \langle a'_{i,j} \rangle_{i \notin D}, tag) :$

$$\bigwedge_{i=1}^L 1 = \text{Sib}_{i-1}.\text{Verify}_1(cpk_{i-1}, \sigma_i, cpk_i, a'_{i,1}, \dots, a'_{i,n_i})$$

$$\wedge 1 = \text{Sib}.\text{Verify}_2(cpk_L, tag, m)\}$$

- Output (TOKEN, sid, at).

Verify

(4) \mathcal{P}_i , upon receiving input (VERIFY, $sid, at, m, \vec{a}_1, \dots, \vec{a}_L$):

- Verify the zero-knowledge proof at with respect to m and $\vec{a}_1, \dots, \vec{a}_L$. Set $f \leftarrow 1$ if valid and $f \leftarrow 0$ otherwise.
- Output (VERIFIED, sid, f).

02 委托证书通用框架

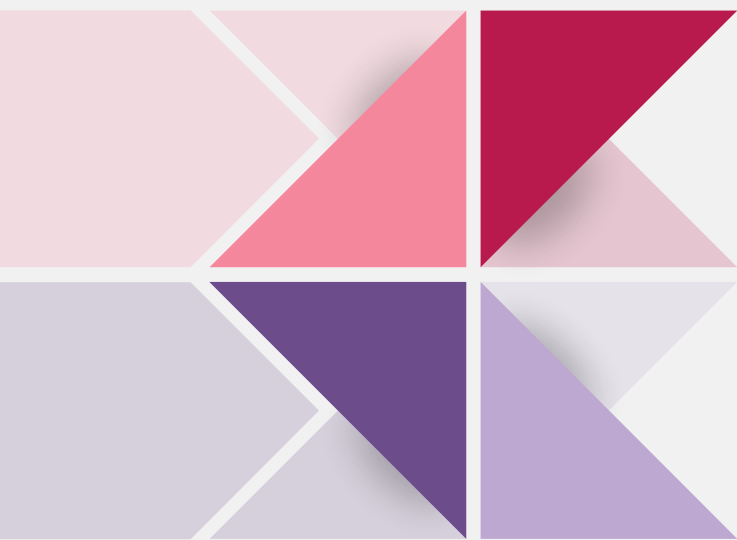
A generic construction for delegatable credentials

通用架构 Π_{dac} 的安全性

THEOREM 4.1. *Our delegatable credentials protocol Π_{dac} securely realizes \mathcal{F}_{dac} (as defined in Section 3), in the $(\mathcal{F}_{\text{smt}}, \mathcal{F}_{\text{ca}}, \mathcal{F}_{\text{crs}})$ -hybrid model, provided that*

- *Sib_t is a secure sibling signature scheme (as defined in Section 2.5),*
- *NIZK is a simulation-sound zero-knowledge proof of knowledge.*

Non-interactive Zero-knowledge



03

使用配对的具体实例

A concrete instantiation using pairings

03 使用配对的具体实例

A concrete instantiation using pairings

A Concrete Proof for the Attribute Tokens

$$\begin{aligned} at \leftarrow \text{NIZK}\{ & (\sigma_1, \dots, \sigma_L, cpk_1, \dots, cpk_L, \langle a'_{i,j} \rangle_{i \notin D}, tag) : \\ & \bigwedge_{i=1,3,\dots}^L 1 = \text{SibGS1.Verify}_1(cpk_{i-1}, \sigma_i, cpk_i, a'_{i,1}, \dots, a'_{i,n_i}) \\ & \bigwedge_{i=2,4,\dots}^L 1 = \text{SibGS2.Verify}_1(cpk_{i-1}, \sigma_i, cpk_i, a'_{i,1}, \dots, a'_{i,n_i}) \\ & \wedge 1 = \text{SibGSb.Verify}_2(cpk_L, tag, m) \} \end{aligned}$$

We propose an efficient instantiation of our generic construction based on the Groth-Schnorr sibling signatures SibGS

03 使用配对的具体实例

A concrete instantiation using pairings

用于生成属性标记的 NIZK 的高效实例化 (下划线标注)

SPK $\{(\langle \underline{s'_i}, \underline{t'_{i,j}} \rangle_{i=1, \dots, L, j=1, \dots, n_i}, \langle a_{i,j} \rangle_{i \notin D}, \langle \underline{cpk_i} \rangle_{i=1, \dots, L-1}, \underline{csk_L}) :$

$$\bigwedge_{i=1,3,\dots}^L \left(e(y_{1,1}, g_2) [e(g_1, ipk)]_{i=1} = e(\underline{s'_i}, r'_i) [e(g_1^{-1}, \underline{cpk_{i-1}})]_{i \neq 1} \wedge \right.$$

$$1_{\mathbb{G}_t} [e(y_{1,1}, ipk)]_{i=1} = e(\underline{t'_{i,1}}, r'_i) [e(\underline{cpk_i}, g_2^{-1})]_{i \neq L} [e(g_1, g_2^{-1})^{\underline{csk_i}}]_{i=L} [e(y_{1,1}^{-1}, \underline{cpk_{i-1}})]_{i \neq 1} \wedge$$

$$\bigwedge_{j:(i,j) \in D} e(a_{i,j}, g_2) [e(y_{1,j+1}, ipk)]_{i=1} = e(\underline{t'_{i,j+1}}, r'_i) [e(y_{1,j+1}^{-1}, \underline{cpk_{i-1}})]_{i \neq 1} \wedge$$

$$\bigwedge_{j:(i,j) \notin D} 1_{\mathbb{G}_t} [e(y_{1,j+1}, ipk)]_{i=1} = e(\underline{t'_{i,j+1}}, r'_i) e(\underline{a_{i,j}}, g_2^{-1}) [e(y_{1,j+1}^{-1}, \underline{cpk_{i-1}})]_{i \neq 1} \bigg) \wedge$$

$$\bigwedge_{i=2,4,\dots}^L \left(e(g_1, y_{2,1}) = e(r'_i, \underline{s'_i}) e(\underline{cpk_{i-1}}, g_2^{-1}) \wedge 1_{\mathbb{G}_t} = e(r'_i, \underline{t'_{i,1}}) e(\underline{cpk_{i-1}}, y_{2,1}^{-1}) [e(g_1^{-1}, \underline{cpk_i})]_{i \neq L} [e(g_1^{-1}, g_2)^{\underline{csk_i}}]_{i=L} \wedge \right.$$

$$\bigwedge_{j:(i,j) \in D} e(g_1, a_{i,j}) = e(r'_i, \underline{t'_{i,j+1}}) e(\underline{cpk_{i-1}}, y_{2,j+1}^{-1}) \wedge \bigwedge_{j:(i,j) \notin D} 1_{\mathbb{G}_t} = e(r'_i, \underline{t'_{i,j+1}}) e(\underline{cpk_{i-1}}, y_{2,j+1}^{-1}) e(g_1^{-1}, \underline{a_{i,j}}) \bigg) \} (sp, r'_1, \dots, r'_L, m).$$

pairing function

- Miller' s algorithm
- exponentiation

Figure 3: Efficient instantiation of the NIZK used to generate attribute tokens (witness underlined for clarity).

03 使用配对的具体实例

A concrete instantiation using pairings

Efficiency analysis

Algorithm	Operations	Total time estimate (ms)
SETUP	$1\{\mathbb{G}_2\}$	1.21
DELEGATE	For each odd Level- i : $1\{\mathbb{G}_2\} + (n_i + 2)\{\mathbb{G}_1\} + (n_i + 1)\{\mathbb{G}_1^2\}$	$2.96 + 1.21n_i$
	For each even Level- i : $1\{\mathbb{G}_1\} + (n_i + 2)\{\mathbb{G}_2\} + (n_i + 1)\{\mathbb{G}_2^2\}$	$5.27 + 3.52n_i$
PRESENT	$\sum_{i=1,3,\dots}^L (1\{\mathbb{G}_2\} + (n_i + 2)\{\mathbb{G}_1\} + (1 + d_i)\{\mathbb{G}_t^2\} + (1 + u_i)\{\mathbb{G}_t^3\} + (2 + n_i)\{\mathbb{G}_1^2\})$	$\sum_{i=1,3,\dots}^L (13.63 + 3.89d_i + 6.11u_i + 1.21n_i) +$
	$\sum_{i=2,4,\dots}^L (1\{\mathbb{G}_1\} + (n_i + 2)\{\mathbb{G}_2\} + (1 + d_i)\{\mathbb{G}_t^2\} + (1 + u_i)\{\mathbb{G}_t^3\} + (2 + n_i)\{\mathbb{G}_2^2\})$	$\sum_{i=2,4,\dots}^L (17.58 + 3.89d_i + 6.11u_i + 3.52n_i)$
VERIFY	$(1 + d_1)E + (3 + u_1 + d_L)E^2 + u_LE^3 + (4 + n_1 + d_L)\{\mathbb{G}_t\} + \sum_{i=2,3,\dots}^{(L-1)} ((1 + d_i)E^2 + (1 + u_i)E^3 + (1 + d_i)\{\mathbb{G}_t\})$	$21.65 + 2.36d_1 + 3.91u_1 + 1.89n_1 + 5.80d_L + 5.48u_L + \sum_{i=2,3,\dots}^{(L-1)} (11.28 + 5.80d_i + 5.48u_i)$

Table 1: Performance evaluation and timing estimations, where d_i and u_i denote the amount of disclosed and undisclosed attributes at delegation level i , respectively, and $n_i = d_i + u_i$; $X\{\mathbb{G}_1^j\}$, $X\{\mathbb{G}_2^j\}$, and $X\{\mathbb{G}_t^j\}$ denote X j -multi-exponentiations in the respective group; $j = 1$ means a simple exponentiation. E^k denote a k -pairing product that we can compute with k -Miller loops and a single shared final exponentiation; $k = 1$ means a single pairing. Benchmarks are (all in ms): $1\{\mathbb{G}_1\} = 0.54$; $1\{\mathbb{G}_1^2\} = 0.67$; $1\{\mathbb{G}_2\} = 1.21$; $1\{\mathbb{G}_2^2\} = 2.31$; $1\{\mathbb{G}_t\} = 1.89$; $1\{\mathbb{G}_t^2\} = 3.89$; $1\{\mathbb{G}_t^3\} = 6.11$; $1E = 2.36$; $1E^2 = 3.91$; $1E^3 = 5.48$.

03 使用配对的具体实例

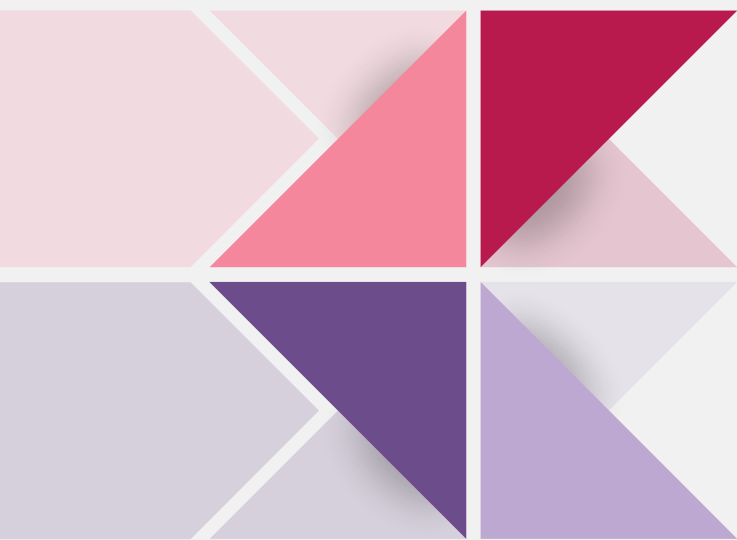
A concrete instantiation using pairings

Environments

C version of Apache Milagro
Cryptographic Library (AMCL)
with a 254-bit Barreto-Naehrig curve
on a 3.1GHz Intel I7-5557U laptop CPU.

n_1	n_2	PRESENT	VERIFY	EST. PRES.	EST. VERIFY
0	0	26.9 ms	20.2 ms	31.21 ms	21.65 ms
1	0	32.7 ms	25.4 ms	38.53 ms	27.45 ms
2	0	38.1 ms	30.9 ms	45.85 ms	33.25 ms
3	0	44.0 ms	36.1 ms	53.17 ms	39.05 ms
4	0	49.5 ms	41.4 ms	60.49 ms	44.85 ms
0	1	38.6 ms	24.8 ms	40.84 ms	27.13 ms
0	2	49.4 ms	29.2 ms	50.47 ms	32.61 ms
0	3	61.5 ms	34.1 ms	60.10 ms	38.09 ms
0	4	72.6 ms	38.7 ms	69.73 ms	43.57 ms
1	1	43.7 ms	30.1 ms	48.16 ms	32.93 ms
2	1	49.3 ms	35.4 ms	55.48 ms	38.73 ms

Table 2: Performance measurements of presenting and verifying Level-2 credentials, and our estimated timings following the computation of Table 1. No attributes are disclosed.



04

许可区块链上的应用

Application to permissioned blockchains

04 许可区块链上的应用

Application to permissioned blockchains

Privacy-Preserving Membership Service

Setup

Certificate Issuance.

Signing Transactions

- 1) signs the transaction content;
- 2) proves a possession of a valid membership credential issued by the CA;
- 3) discloses the attributes that are required by the access control policy for the transaction.

To enable certificate revocation and auditing, the token can also prove in zero-knowledge:

- 4) that the certificate was not revoked with respect to the revocation information published by the membership service (or a designated revocation authority);
- 5) provide a ciphertext that contains the credential identifier encrypted under the auditor's public encryption key (so that only the auditor can decrypt it) and a ZK proof that the same credential identifier is contained in the membership certificate, without disclosing the identifier itself.

04 许可区块链上的应用

Application to permissioned blockchains

Hierarchical Membership Service from Delegatable Credentials with Attributes



Organizations(CAs)



parties that are allowed
to submit transactions

04 许可区块链上的应用

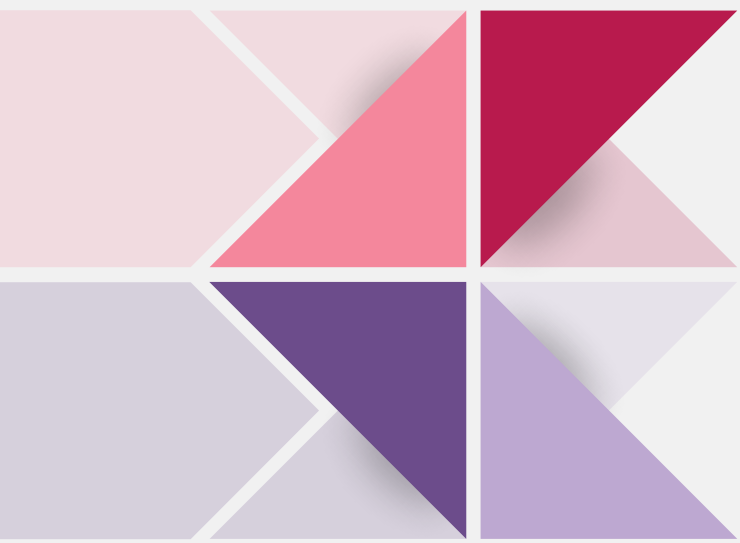
Applicataion to permissioned blockchains

Implementation and Performance Analysis

C version of Apache Milagro
Cryptographic Library (AMCL)
with a 254-bit Barreto-Naehrig curve
on a 3.1GHz Intel I7-5557U laptop CPU.

n_1	n_2	PRESENT	VERIFY	EST. PRES.	EST. VERIFY
0	0	26.9 ms	20.2 ms	31.21 ms	21.65 ms
1	0	32.7 ms	25.4 ms	38.53 ms	27.45 ms
2	0	38.1 ms	30.9 ms	45.85 ms	33.25 ms
3	0	44.0 ms	36.1 ms	53.17 ms	39.05 ms
4	0	49.5 ms	41.4 ms	60.49 ms	44.85 ms
0	1	38.6 ms	24.8 ms	40.84 ms	27.13 ms
0	2	49.4 ms	29.2 ms	50.47 ms	32.61 ms
0	3	61.5 ms	34.1 ms	60.10 ms	38.09 ms
0	4	72.6 ms	38.7 ms	69.73 ms	43.57 ms
1	1	43.7 ms	30.1 ms	48.16 ms	32.93 ms
2	1	49.3 ms	35.4 ms	55.48 ms	38.73 ms

Table 2: Performance measurements of presenting and verifying Level-2 credentials, and our estimated timings following the computation of Table 1. No attributes are disclosed.



05

论文总结
Conclusion

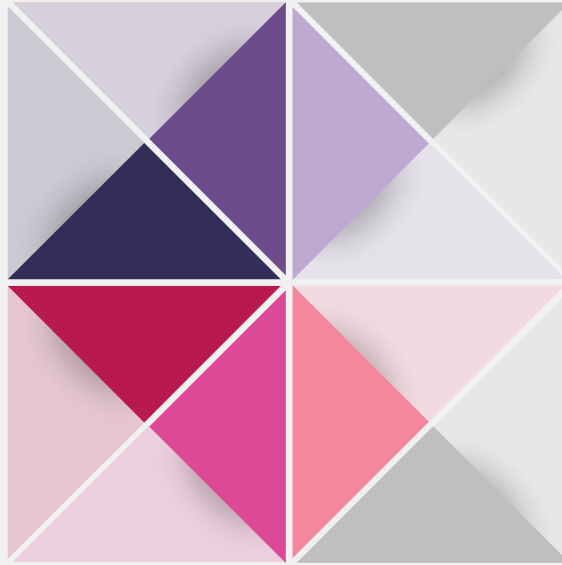
05 论文总结

Conclusion

Conclusion

The first practical delegatable credential system with attributes presented in this paper addresses the basic privacy and security needs of a public key infrastructure and, in particular, the requirements of a membership service of a permissioned blockchain. However, there are a number of additional functionalities that could be considered, such as key life cycle management, revocation, and support for auditable tokens. We expect that the solutions for these extensions known for the ordinary anonymous credentials to be applicable here as well. Of course, any of these extensions would require to modify our ideal functionality \mathcal{F}_{dac} , as would the extension of a distributed issuance of Level-1 credentials. One way to do it is to extend our ideal functionality \mathcal{F}_{dac} to accept as input and also output commitments to attribute values, following the recent work by Camenisch et al. [7]. This would allow for a modular construction of a delegatable credential scheme with the extensions just discussed. We consider all of this future work.





Thank you