Anonymous Transactions with Revocation and Auditing in Hyperledger Fabric

Anonymous Credentials, Revocation, Auditing, Blockchain

DOI: 10.1007/978-3-030-92548-2_23

Dmytro Bogatov, Angelo De Caro, Kaoutar Elkhiyaoui, Björn Tackmann
dmytro@bu.edu, adc@zurich.ibm.com, kao@zurich.ibm.com
bjoern@dfinity.org

Built from bc167548 on December 25, 2021

Boston University
Graduate School of Arts and Sciences
Department of Computer Science
BACKGROUND
Motivation

- Permissioned blockchains explicitly register their participants logistics network, bank transactions, know-your-customer and anti-money-laundering regulations
- Prove the permission to post a transaction without revealing identity
  - arbitrary length of delegation chain
  - dynamically embedding an arbitrary number of attributes
  - efficient and integrated with the blockchain
- Revocation and audit
  - Prove \( \text{AGE} \geq 21 \) using driving license without revealing your name and issuing state delegatable anonymous credentials
  - Allow a state to suspend (revoke) the driving license at any time privacy-preserving revocation
  - Allow a federal investigator to inspect the entire license at a later date with a warrant auditing along with proving that the credential is “auditable”
Motivation

• Permissioned blockchains explicitly register their participants logistics network, bank transactions, know-your-customer and anti-money-laundering regulations

• Prove the permission to post a transaction without revealing identity
  • arbitrary length of delegation chain
  • dynamically embedding an arbitrary number of attributes
  • efficient and integrated with the blockchain

• Revocation and audit
  • Prove AGE ≥ 21 using driving license without revealing your name and issuing state delegatable anonymous credentials
  • Allow a state to suspend (revoke) the driving license at any time privacy-preserving revocation
  • Allow a federal investigator to inspect the entire license at a later date with a warrant auditing along with proving that the credential is “auditable”
Motivation

• Permissioned blockchains explicitly register their participants logistics network, bank transactions, know-your-customer and anti-money-laundering regulations
• Prove the permission to post a transaction without revealing identity
  • arbitrary length of delegation chain
  • dynamically embedding an arbitrary number of attributes
  • efficient and integrated with the blockchain
• Revocation and audit
  • Prove $\text{AGE} \geq 21$ using driving license without reveling your name and issuing state delegatable anonymous credentials
  • Allow a state to suspend (revoke) the driving license at any time privacy-preserving revocation
  • Allow a federal investigator to inspect the entire license at a later date with a warrant auditing along with proving that the credential is “auditable”
Motivation

- Permissioned blockchains explicitly register their participants logistics network, bank transactions, know-your-customer and anti-money-laundering regulations
- Prove the permission to post a transaction without revealing identity
  - arbitrary length of delegation chain
  - dynamically embedding an arbitrary number of attributes
  - efficient and integrated with the blockchain
- Revocation and audit
  - Prove $\text{AGE} \geq 21$ using driving license without reveling your name and issuing state delegatable anonymous credentials
  - Allow a state to suspend (revoke) the driving license at any time privacy-preserving revocation
  - Allow a federal investigator to inspect the entire license at a later date with a warrant auditing along with proving that the credential is “auditable”
Motivation

- Permissioned blockchains explicitly register their participants logistics network, bank transactions, know-your-customer and anti-money-laundering regulations
- Prove the permission to post a transaction without revealing identity
  - arbitrary length of delegation chain
  - dynamically embedding an arbitrary number of attributes
  - efficient and integrated with the blockchain
- Revocation and audit
  - Prove \( \text{AGE} \geq 21 \) using driving license without reveling your name and issuing state delegatable anonymous credentials
  - Allow a state to suspend (revoke) the driving license at any time privacy-preserving revocation
  - Allow a federal investigator to inspect the entire license at a later date with a warrant auditing along with proving that the credential is “auditable”
Components

Clients invoke TXs and observe their results

Peers execute and validate TXs

Orderers determine the order of TXs and distribute the blocks to peers

MSP maintains & manages IDs of all members

Execute-Order-Validate

- Client sends TX proposal to endorsers
- Endorsers execute TX, sign read/write sets
- Client prepares TX, sends to orderers
- Orderers puts TX in block, distributes
- All peers validate TX
Components

Clients invoke TXs and observe their results

Peers execute and validate TXs

Orderers determine the order of TXs and distribute the blocks to peers

MSP maintains & manages IDs of all members

Execute-Order-Validate

- Client sends TX proposal to endorsers
- Endorsers execute TX, sign read/write sets
- Client prepares TX, sends to orderers
- Orderers puts TX in block, distributes
- All peers validate TX
Delegatable Anonymous Credentials [7]

- **KEYGEN**(sp) \(\rightarrow\) (csk, cpk)
generate a pair of keys for the caller (root, intermediate, user)
- **DELEGATE**(cski, credi, cpki+1, \(\vec{a}_{i+1}\)) \(\rightarrow\) credi+1
  Level-i authority produces credentials of the Level-\((i + 1)\)
  binding attributes \(\vec{a}_{i+1}\) to public key cpki+1
- **PRESENT**(cskl, credl, cpk0, \(\langle a_{i,j}\rangle_{(i,j)\in D}, m\)) \(\rightarrow\) \(P_{\text{cred}}\)
  shows the validity of credl under cpk0, proves that secret key cskl matches credl and disclosed attributes \(\langle a_{i,j}\rangle_{(i,j)\in D}\), signs \(m\)
- **VERIFY**(\(P_{\text{cred}}\), cpk0, \(\langle a_{i,j}\rangle_{(i,j)\in D}, m\)) \(\rightarrow\) \{0, 1\}
  verifies the correctness of \(P_{\text{cred}}\) relative to disclosed attributes \(\langle a_{i,j}\rangle_{(i,j)\in D}\), message \(m\) and public key cpk0
Delegatable Anonymous Credentials [7]

• **KEYGEN**(sp) $\rightarrow_{s}$ (csk, cpk)
  
  generate a pair of keys for the caller (root, intermediate, user)

• **DELEGATE**(cski, credi, cpki+1, $\vec{a}_{i+1}$) $\rightarrow_{s}$ credi+1
  
  Level-i authority produces credentials of the Level-($i + 1$)
  binding attributes $\vec{a}_{i+1}$ to public key cpki+1

• **PRESENT**(cskl, credL, cpk0, $\langle a_{i,j} \rangle_{(i,j) \in D}, m$) $\rightarrow_{s}$ $\mathcal{P}_{\text{cred}}$
  
  shows the validity of credL under cpk0, proves that secret key
  cskL matches credL and disclosed attributes $\langle a_{i,j} \rangle_{(i,j) \in D}$, signs m

• **VERIFY**(Pcred, cpk0, $\langle a_{i,j} \rangle_{(i,j) \in D}, m$) $\rightarrow \{0, 1\}$
  
  verifies the correctness of $\mathcal{P}_{\text{cred}}$ relative to disclosed attributes
  $\langle a_{i,j} \rangle_{(i,j) \in D}$, message m and public key cpk0

• Instantiated with Groth [5] and Schnorr [1] signature
  schemes in [7]
Delegatable Anonymous Credentials [7]

- **KEYGEN**(sp) → (csk, cpk)
generate a pair of keys for the caller (root, intermediate, user)

- **DELEGATE**(cski, credi, cpki+1, āi+1) → credi+1

  Level-i authority produces credentials of the Level-((i + 1)

  binding attributes āi+1 to public key cpki+1

- **PRESENT**(cskL, credL, cpk0, ⟨ai,j⟩(i,j)∈D, m) → ℙcred

  shows the validity of credL under cpk0, proves that secret key
  cskL matches credL and disclosed attributes ⟨ai,j⟩(i,j)∈D, signs m

- **VERIFY**(ℙcred, cpk0, ⟨ai,j⟩(i,j)∈D, m) → {0, 1}

  verifies the correctness of ℙcred relative to disclosed attributes
  ⟨ai,j⟩(i,j)∈D, message m and public key cpk0

Delegatable Anonymous Credentials [7]

- **KEYGEN**(sp) → (csk, cpk)
generate a pair of keys for the caller (root, intermediate, user)

- **DELEGATE**(csk_i, cred_i, cpk_i+1, a_i+1) → cred_i+1
  Level-i authority produces credentials of the Level-(i + 1)
binding attributes a_i+1 to public key cpk_i+1

- **PRESENT**(csk_L, cred_L, cpk_0, ⟨a_{i,j}⟩_{(i,j)∈D}, m) → \(\mathcal{P}_{\text{cred}}\)
  shows the validity of cred_L under cpk_0, proves that secret key
csk_L matches cred_L and disclosed attributes ⟨a_{i,j}⟩_{(i,j)∈D}, signs m

- **VERIFY**(\(\mathcal{P}_{\text{cred}}\), cpk_0, ⟨a_{i,j}⟩_{(i,j)∈D}, m) → \{0, 1\}
  verifies the correctness of \(\mathcal{P}_{\text{cred}}\) relative to disclosed attributes
  ⟨a_{i,j}⟩_{(i,j)∈D}, message m and public key cpk_0

Delegatable Anonymous Credentials [7]

- **KEYGEN**(sp) $\rightarrow_s$ (csk, cpk)
generate a pair of keys for the caller (root, intermediate, user)

- **DELEGATE**(cski, credi, cpki+1, $\bar{a}_{i+1}$) $\rightarrow_s$ credi+1
  Level-i authority produces credentials of the Level-$(i + 1)$
  binding attributes $\bar{a}_{i+1}$ to public key cpki+1

- **PRESENT**(cskL, credL, cpk0, $\langle a_{i,j}\rangle_{(i,j)\in D}$, m) $\rightarrow_s$ $\Psi_{cred}$
  shows the validity of credL under cpk0, proves that secret key
  cskL matches credL and disclosed attributes $\langle a_{i,j}\rangle_{(i,j)\in D}$, signs m

- **VERIFY**( $\Psi_{cred}$, cpk0, $\langle a_{i,j}\rangle_{(i,j)\in D}$, m) $\rightarrow$ {0, 1}
  verifies the correctness of $\Psi_{cred}$ relative to disclosed attributes
  $\langle a_{i,j}\rangle_{(i,j)\in D}$, message m and public key cpk0

Delegatable Anonymous Credentials [7]

- **KEYGEN**(sp) → (csk, cpk)
  generate a pair of keys for the caller (root, intermediate, user)

- **DELEGATE**(csk_i, cred_i, cpk_{i+1}, \(\vec{a}_{i+1}\)) → cred_{i+1}
  Level-i authority produces credentials of the Level-(i + 1) binding attributes \(\vec{a}_{i+1}\) to public key cpk_{i+1}

- **PRESENT**(csk_L, cred_L, cpk_0, \(\langle a_{i,j}\rangle_{(i,j)\in D}, m\)) → \(\mathcal{P}_{\text{cred}}\)
  shows the validity of cred_L under cpk_0, proves that secret key csk_L matches cred_L and disclosed attributes \(\langle a_{i,j}\rangle_{(i,j)\in D}\), signs m

- **VERIFY**(\(\mathcal{P}_{\text{cred}}, cpk_0, \langle a_{i,j}\rangle_{(i,j)\in D}, m\)) → \{0, 1\}
  verifies the correctness of \(\mathcal{P}_{\text{cred}}\) relative to disclosed attributes \(\langle a_{i,j}\rangle_{(i,j)\in D}\), message m and public key cpk_0

IMPROVED CONSTRUCTION
Revocation

General approach

- Revocation is inherently at odds with anonymity
- We couple *epoch-based whitelisting* with signatures in a way that yields efficient proofs of non-revocation
- *Epochs* defined in terms of blockchain height
- *Epoch handle* (signature) binds public key to epoch

Instantiation

- Simple way: epoch as an attribute
- Explicit proof of non-revocation allows to decouple credential issuer and revocation authority

\[
\varepsilon := g^{\text{HASH}(\text{epoch})}
\]

\[
\sigma \leftarrow \text{GROTH.SIGN}(\text{rsk}; \varepsilon, \text{cpk})
\]

\[
\begin{align*}
\mathcal{P} \leftarrow & \text{NIZK}\{(\sigma_1, \ldots, \sigma_L, \text{cpk}_1, \ldots, \text{cpk}_L, \langle a_{i,j} \rangle_{(i,j) \notin D}, \sigma_m, \sigma) : \\
& \quad \bigwedge_{i=2,4,\ldots,L} \text{GROTH1.VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \\
& \quad \bigwedge_{i=1,3,\ldots,L} \text{GROTH2.VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \\
& \quad \bigwedge \text{SCHNORR.VERIFY}(\text{cpk}_L; \sigma_m; m) \\
& \quad \bigwedge \text{GROTH.VERIFY}(\text{rpk}; \sigma; \varepsilon, \text{cpk}_L) \}
\end{align*}
\]
General approach

- Revocation is inherently at odds with anonymity
- We couple *epoch-based whitelisting* with signatures in a way that yields efficient proofs of non-revocation
- *Epochs* defined in terms of blockchain height
- *Epoch handle* (signature) binds public key to epoch

Instantiation

- Simple way: epoch as an attribute
- Explicit proof of non-revocation allows to decouple credential issuer and revocation authority

\[
\varepsilon := g^{\text{HASH}(\text{epoch})} \\
\sigma \leftarrow \text{GROTH.SIGN}(\text{rsk}; \varepsilon, \text{cpk})
\]

\[
\Psi \leftarrow \text{NIZK}\{(\sigma_1, \ldots, \sigma_L, \text{cpk}_1, \ldots, \text{cpk}_L, \langle a_{i,j} \rangle_{(i,j) \notin D}, \sigma_m, \sigma) : \\
\land_{i=2,4,\ldots,L} \text{GROTH}1.\text{VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \\
\land_{i=1,3,\ldots,L} \text{GROTH}2.\text{VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \\
\land \text{SCHNORR.VERIFY}(\text{cpk}_L; \sigma_m; m) \\
\land \text{GROTH.VERIFY}(\text{rpk}; \sigma; \varepsilon, \text{cpk}_L) \}
\]
Revocation

General approach

- Revocation is inherently at odds with anonymity
- We couple *epoch-based whitelisting* with signatures in a way that yields efficient proofs of non-revocation
- *Epochs* defined in terms of blockchain height
- *Epoch handle* (signature) binds public key to epoch

\[ \mathcal{P} \leftrightarrow \text{NIZK}\{ (\sigma_1, \ldots, \sigma_L, \text{cpk}_1, \ldots, \text{cpk}_L, \langle a_{i,j} \rangle_{(i,j) \notin D}, \sigma_m, \sigma) : \]
\[ \bigwedge_{i=2,4,\ldots,L} \text{GROTH}_1.\text{VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \]
\[ \bigwedge_{i=1,3,\ldots,L} \text{GROTH}_2.\text{VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \]

\[ \wedge \text{SCHNORR.VERIFY}(\text{cpk}_L; \sigma_m; m) \]
\[ \wedge \text{GROTH.VERIFY}(\text{rpk}; \sigma; \varepsilon, \text{cpk}_L) \} \]

Instantiation

- Simple way: epoch as an attribute
- Explicit proof of non-revocation allows to decouple credential issuer and revocation authority

\[ \varepsilon := g^{\text{HASH}(\text{epoch})} \]
\[ \sigma \leftrightarrow \text{GROTH.SIGN}(\text{rsk}; \varepsilon, \text{cpk}) \]

\[ \text{GROTH.VERIFY}(\text{rpk}; \sigma; \varepsilon, \text{cpk}_L) \]
Audit

General approach

- TX author embeds her ID (public key) encrypted under the auditor’s public key
- Prove that the user encrypts her own public key and uses the public key of the authorized auditor

Instantiation

- User generates ElGamal secret and public keys (ask, \(apk = g^{ask}\))
- Encerts with enc = \((cpk_L \cdot apk^\rho, g^\rho)\)
- Auditor decrypts enc guaranteed to succeed

\[\Psi \leftarrow NIZK\{\left(\pi_{L,1}, \ldots, \pi_{L,L}, \pi_{1,1}, \ldots, \pi_{1,n}\right) : \]
\[\bigwedge_{i=2,4,\ldots}^L \text{GROTH}_1.\text{VERIFY}(cpk_{i-1}, \pi_i, cpk_i, a_{1,1}, \ldots, a_{i,n}) \]
\[\bigwedge_{i=1,3,\ldots}^L \text{GROTH}_2.\text{VERIFY}(cpk_{i-1}, \pi_i, cpk_i, a_{1,1}, \ldots, a_{i,n}) \]
\[\wedge \text{SCHNORR.VERIFY}(cpk_L, \pi_m, m) \]
\[\wedge \text{GROTH.VERIFY}(rpk, \pi, \epsilon, cpk_L) \]
\[\wedge \text{enc} = (cpk_L \cdot apk^\rho, g^\rho) \]
Audit

General approach

- TX author embeds her ID (public key) encrypted under the auditor’s public key
- Prove that the user encrypts her own public key and uses the public key of the authorized auditor

Instantiation

- User generates ElGamal secret and public keys (ask, apk = $g^{ask}$)
- Encrypts with $\text{enc} = (\text{cpk}_L \cdot \text{apk}^\rho, g^\rho)$
- Auditor decrypts enc guaranteed to succeed

\[ \mathcal{P} \leftarrow \text{NIZK}\{(\sigma_1, \ldots, \sigma_L, \text{cpk}_1, \ldots, \text{cpk}_L, (a_{i,j})_{(i,j) \notin D}, \sigma_m, \sigma, \rho) : \right. \]

\[ \bigwedge_{i=2,4,\ldots}^L \text{GROTH}_1.\text{VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \]

\[ \bigwedge_{i=1,3,\ldots}^L \text{GROTH}_2.\text{VERIFY}(\text{cpk}_{i-1}; \sigma_i; \text{cpk}_i, a_{i,1}, \ldots, a_{i,n_i}) \]

\[ \bigwedge \text{SCHNORR.VERIFY}(\text{cpk}_L; \sigma_m; m) \]

\[ \bigwedge \text{GROTH.VERIFY}(\text{rpk}; \sigma; \varepsilon, \text{cpk}_L) \]

\[ \text{enc} = (\text{cpk}_L \cdot \text{apk}^\rho, g^\rho) \} \]
Audit

General approach

- TX author embeds her ID (public key) encrypted under the auditor’s public key
- Prove that the user encrypts her own public key and uses the public key of the authorized auditor

**Instantiation**

- User generates ElGamal secret and public keys ($ask, apk = g^{ask}$)
- Encrypts with $enc = (cpk_L \cdot apk^\rho, g^\rho)$
- Auditor decrypts $enc$ guaranteed to succeed

\[
\Psi \leftarrow \text{NIZK}\{ (\sigma_1, \ldots, \sigma_L, cpk_{1, \ldots, L}, \langle a_{ij} \rangle_{i,j} \notin D, \sigma_m, \sigma, \rho) : \\
\bigwedge_{i=2,4,\ldots}^L \text{GROTH}_1.\text{VERIFY}(cpk_{i-1}; \sigma_i; cpk_i, a_{i,1}, \ldots, a_{i,n_i}) \\
\bigwedge_{i=1,3,\ldots}^L \text{GROTH}_2.\text{VERIFY}(cpk_{i-1}; \sigma_i; cpk_i, a_{i,1}, \ldots, a_{i,n_i}) \\
\land \text{SCHNORR}.\text{VERIFY}(cpk_L; \sigma_m; m) \\
\land \text{GROTH}.\text{VERIFY}(rpk; \varepsilon, cpk_L) \\
\land enc = (cpk_L \cdot apk^\rho, g^\rho) \}
\]
Performance optimizations

- Simplified the pseudocode in [7] and corrected mistakes
- Parallelized on the granularity of commitments
- Optimal Miller’s loop and final exponentiation

\[
\prod_i e(a_i, b_i)^{c_i} = \text{FEXP} \left( \prod_i \hat{t}(a_i^{c_i}, b_i) \right) = \text{FEXP} \left( \prod_i \hat{t}(a_i, b_i^{c_i}) \right)
\]

Require: \(a_i \in G_1, b_i \in G_2, c_i \in \mathbb{Z}_q \cup \perp\) for \(L = 1, \ldots, n\)
Ensure: \(EPRODUCT((a_i, b_i, c_i)_{i=1}^n) = \prod_{i=1}^n e(a_i, b_i)^{c_i}\)

1: procedure \(EPRODUCT((a_i, b_i, c_i)_{i=1}^n)\)
2: \(r := 1_T \in G_T \triangleright\) an identity element
3: for \(i = (1, \ldots, n)\) do
4: \(\quad\) if \(c_i \neq \perp\) then
5: \(\quad\) \(a_i := a_i^{c_i}\)
6: \(\quad\) for \(i = (1, 3, \ldots, n)\) do
7: \(\quad\quad\) if \(a_{i+1} \neq \perp\) then
8: \(\quad\quad\quad\) \(\triangleright\hat{t}_2\) is a more efficient version of \(\hat{t} \cdot \hat{t}\)
9: \(\quad\quad\) \(r := r \cdot \hat{t}_2 (a_i, b_i, a_{i+1}, b_{i+1})\)
10: \(\quad\) else
11: \(\quad\) \(r := r \cdot \hat{t} (a_i, b_i)\)
12: return \(\text{FEXP} (r)\)
Optimizations

Performance optimizations

- Simplified the pseudocode in [7] and corrected mistakes
- Parallelized on the granularity of commitments
- Optimal Miller’s loop and final exponentiation

\[
\prod_i e(a_i, b_i)^{c_i} = \text{FEXP} \left( \prod_i \hat{t}(a_i^{c_i}, b_i) \right) = \text{FEXP} \left( \prod_i \hat{\hat{t}}(a_i, b_i^{c_i}) \right)
\]

Require: \( a_i \in G_1, b_i \in G_2, c_i \in \mathbb{Z}_q \cup \perp \) for \( L = 1, \ldots, n \)
Ensure: \( \text{EPRODUCT}(\langle a_i, b_i, c_i \rangle_{i=1}^n) = \prod_{i=1}^n e(a_i, b_i)^{c_i} \)

1: procedure \( \text{EPRODUCT}(\langle a_i, b_i, c_i \rangle_{i=1}^n) \)
2: \( r := 1_T \in G_T \quad \triangleright \) an identity element
3: for \( i = (1, \ldots, n) \) do
4: if \( c_i \neq \perp \) then
5: \( a_i := a_i^{c_i} \)
6: for \( i = (1, 3, \ldots, n) \) do
7: if \( a_{i+1} \neq \perp \) then
8: \( \triangleright \hat{\hat{t}}_2 \) is a more efficient version of \( \hat{t} \cdot \hat{t} \)
9: \( r := r \cdot \hat{\hat{t}}_2(a_i, b_i, a_{i+1}, b_{i+1}) \)
10: else
11: \( r := r \cdot \hat{t}(a_i, b_i) \)
12: return \( \text{FEXP}(r) \)
Optimizations

Performance optimizations

- Simplified the pseudocode in [7] and corrected mistakes
- Parallelized on the granularity of commitments
- Optimal Miller’s loop and final exponentiation

\[ \prod_i e(a_i, b_i)^{c_i} = \text{FEXP} \left( \prod_i \hat{t}(a_i^{c_i}, b_i) \right) = \text{FEXP} \left( \prod_i \hat{t}(a_i, b_i^{c_i}) \right) \]

Require: \( a_i \in \mathbb{G}_1, b_i \in \mathbb{G}_2, c_i \in \mathbb{Z}_q \cup \perp \) for \( L = 1, \ldots, n \)

Ensure: \( \text{EPRODUCT}((a_i, b_i, c_i)_{i=1}^n) = \prod_{i=1}^n e(a_i, b_i)^{c_i} \)

1: procedure \( \text{EPRODUCT}((a_i, b_i, c_i)_{i=1}^n) \)
2: \( r := 1_T \in \mathbb{G}_T \) \( \triangleright \) an identity element
3: for \( i = (1, \ldots, n) \) do
4: \hspace{1em} if \( c_i \neq \perp \) then
5: \hspace{2em} \( a_i := a_i^{c_i} \)
6: for \( i = (1, 3, \ldots, n) \) do
7: \hspace{2em} if \( a_{i+1} \neq \perp \) then
8: \hspace{3em} \( \triangleright \hat{t}_2 \) is a more efficient version of \( \hat{t} \cdot \hat{t} \)
9: \hspace{3em} \( r := r \cdot \hat{t}_2 (a_i, b_i, a_{i+1}, b_{i+1}) \)
10: \hspace{2em} else
11: \hspace{3em} \( r := r \cdot \hat{t} (a_i, b_i) \)
12: return \( \text{FEXP} (r) \)
Integration with Fabric. Setup.

Setup

1: Level-\(i\) CA

\[
\text{.. \ldots Repeated for } L \text{ rounds of delegation (from the Root CA to Intermediate CAs to the User) \ldots \ldots}
\]

2: \(\text{csk}_i \leftarrow \mathbb{Z}_q, \text{cpk}_i := g^{\text{csk}_i}\)

3: \(\text{nonce} \leftarrow \{0, 1\}^\lambda\)

4: \(\text{VERIFYPK}(\mathcal{P}_{\text{pk}}, \text{cpk}_{i+1}, \text{nonce})\)

5: \(\sigma_{i+1} \leftarrow \mathcal{GROTH}.\text{SIGN}(\text{csk}_i; \text{cpk}_{i+1}, \vec{a}_{i+1})\)

\(\text{Level-(}i+1\text{) CA}\)

\(\text{csk}_{i+1} \leftarrow \mathbb{Z}_q, \text{cpk}_{i+1} := f^{\text{csk}_{i+1}}\)

\(\mathcal{P}_{\text{pk}} \leftarrow \mathcal{PROVEPK}(\text{csk}_{i+1}, \text{cpk}_{i+1}, \text{nonce})\)

\(\sigma_{i+1}\)
Integration with Fabric. Revocation.

Revocation

1: Revocation authority

\[ \text{On each epoch, user requests a non-revocation handle} \]

2: \( rsk \leftarrow \mathbb{Z}_q, \ rpk := g^{rsk} \quad \text{csk} \leftarrow \mathbb{Z}_q, \ cpk := g^{cks} \)

3: \( \text{nonce} \leftarrow \{0, 1\}^\lambda \)

4: \( \text{VERIFYPK}(\mathcal{P}_{pk}, \ cpk, \ \text{nonce}) \)

5: \( \sigma \leftarrow \mathcal{NRSIGN}(rsk, \ cpk, \ epoch) \)

\[ \sigma, \ \text{epoch} \]
Integration with Fabric. Transaction submission.

Transaction submission

1: Verifier

2: User

cred := (⟨σ_j, a_j, cpk_j⟩_{j=1})

User submits a transaction

3: User submits a transaction

4: enc, ρ := AUDITENC(apk, cpk)

5: sk_nym, pk_nym ← MAKENYM(csk)

6: ϕ_rev ← NRPROVE(σ, csk, sk_nym, epoch)

7: ϕ_audit ← AUDITPROVE(enc, ρ, cpk, csk, pk_nym, sk_nym)

8: ϕ_cred ← CREDPROVE(cred, D, sk_nym, csk, ⊥)

9: ϕ_nym ← SIGNNYM(pk_nym, sk_nym, csk, tx)

10: m, σ_nym := (ϕ_cred, ϕ_rev, ϕ_audit, enc, tx, pk_nym)

11: VERIFYNYM(pk_nym, tx, σ_nym)

12: NRVERIFY(ϕ_rev, pk_nym, epoch)

13: AUDITVERIFY(ϕ_audit, enc, pk_nym)

14: CREDVERIFY(ϕ_cred, D, pk_nym, ⊥)
EXPERIMENTAL EVALUATION
• Implemented a stand-alone production-ready library in Go [9]
  470 tests cover 100% of the code
• Benchmarks run on **c2-standard-60** GCE VM running Ubuntu 18.04
  all benchmarked operations were run 100 times
• Default number of levels and attributes per level are $L = 2$ and $n = 2$
• Implemented a distributed prototype of Fabric using our scheme
• Setup is different from the base scheme in [7]
• Aiming to answer 6 evaluation questions
Question 1: what is the optimizations’ performance benefit?

<table>
<thead>
<tr>
<th>e-product</th>
<th>Parallelization</th>
<th>CREDPROVE</th>
<th>CREDVERIFY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Big</td>
<td>Small</td>
<td>Big</td>
</tr>
<tr>
<td>disabled</td>
<td>disabled</td>
<td>2 873</td>
<td>843</td>
</tr>
<tr>
<td>enabled</td>
<td>disabled</td>
<td>1 312</td>
<td>341</td>
</tr>
<tr>
<td>disabled</td>
<td>enabled</td>
<td>1 480</td>
<td>357</td>
</tr>
<tr>
<td>enabled</td>
<td>enabled</td>
<td>890</td>
<td>191</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Improvement (≈ times)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>4.4</td>
<td>3.9</td>
<td>4.8</td>
</tr>
</tbody>
</table>

Optimizations benchmark for $L = 2$ and $n = 2$ (small) and $L = 5$ and $n = 3$ (big). The values are in milliseconds.
Question 2: how does the scheme scale with the number of levels and attributes?

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>41 ms</td>
<td>51 ms</td>
<td>63 ms</td>
<td>72 ms</td>
<td>82 ms</td>
</tr>
<tr>
<td>1</td>
<td>89 ms</td>
<td>110 ms</td>
<td>116 ms</td>
<td>153 ms</td>
<td>173 ms</td>
</tr>
<tr>
<td></td>
<td>398 B</td>
<td>534 B</td>
<td>670 B</td>
<td>806 B</td>
<td>942 B</td>
</tr>
<tr>
<td>2</td>
<td>94 ms</td>
<td>138 ms</td>
<td>192 ms</td>
<td>255 ms</td>
<td>315 ms</td>
</tr>
<tr>
<td></td>
<td>801 B</td>
<td>1.2 kB</td>
<td>1.6 kB</td>
<td>2.0 kB</td>
<td>2.4 kB</td>
</tr>
<tr>
<td>3</td>
<td>173 ms</td>
<td>273 ms</td>
<td>367 ms</td>
<td>516 ms</td>
<td>616 ms</td>
</tr>
<tr>
<td></td>
<td>1.2 kB</td>
<td>1.7 kB</td>
<td>2.3 kB</td>
<td>2.8 kB</td>
<td>3.3 kB</td>
</tr>
<tr>
<td>4</td>
<td>333 ms</td>
<td>542 ms</td>
<td>661 ms</td>
<td>891 ms</td>
<td>1 146 ms</td>
</tr>
<tr>
<td>5</td>
<td>276 ms</td>
<td>342 ms</td>
<td>391 ms</td>
<td>500 ms</td>
<td>648 ms</td>
</tr>
<tr>
<td></td>
<td>2.0 kB</td>
<td>2.9 kB</td>
<td>3.9 kB</td>
<td>4.8 kB</td>
<td>5.7 kB</td>
</tr>
</tbody>
</table>

Parameters benchmark. In each cell the top value is a proof generation overhead, the middle value is a proof verification overhead and the bottom value is the proof size.
Question 3: what overhead do our extensions impose?

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Time $G_1$</th>
<th>Time $G_2$</th>
<th>Procedure</th>
<th>Time $G_1$</th>
<th>Time $G_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROTH.KEYGEN</td>
<td>1.6</td>
<td>4.7</td>
<td>GROTH.SIGN</td>
<td>16</td>
<td>41</td>
</tr>
<tr>
<td>GROTH.RANDOMIZE</td>
<td>11</td>
<td>23</td>
<td>GROTH.VERIFY</td>
<td>53</td>
<td>62</td>
</tr>
<tr>
<td>SCHRNNORR.SIGN</td>
<td>1.6</td>
<td>4.8</td>
<td>SCHRNNORR.VERIFY</td>
<td>2</td>
<td>9.6</td>
</tr>
<tr>
<td>AUDITENCRYPT</td>
<td>3</td>
<td>9.4</td>
<td>NRSIGN</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>AUDITPROVE</td>
<td>5.8</td>
<td>24</td>
<td>NRPROVE</td>
<td>66</td>
<td>88</td>
</tr>
<tr>
<td>AUDITVERIFY</td>
<td>9.2</td>
<td>39</td>
<td>NRVERIFY</td>
<td>127</td>
<td>149</td>
</tr>
<tr>
<td>MAKEnym</td>
<td>2.1</td>
<td>9.4</td>
<td>PROVEPK</td>
<td>3.1</td>
<td>9.4</td>
</tr>
<tr>
<td>SIGNnym</td>
<td>2.2</td>
<td>9.9</td>
<td>VERIFYPK</td>
<td>2</td>
<td>9.5</td>
</tr>
<tr>
<td>VERIFYnym</td>
<td>3.5</td>
<td>14</td>
<td>KEYGEN</td>
<td>1.5</td>
<td>4.2</td>
</tr>
</tbody>
</table>

Running time of extensions in milliseconds.
Question 4: how does the system compare to the old non-delegatable idemix?

- Ran workloads against current (non-delegatable) Fabric idemix and regular MSP generating secrets, signing and verifying identities
- Idemix in Fabric \([6]\) uses BBS+ signatures \([3]\)
  - ran actual Fabric code
- Default (not-idemix) Fabric MSP simply uses X.509 certificates and ECDSA algorithms \([4]\)
  - ran ECDSA routines in Go `crypto` module using P-384 curve — most secure option in Fabric
- Results show the relative costs of using more privacy-preserving solutions
  - default MSP takes 21 ms
  - idemix MSP in Fabric takes 108 ms
  - our solution takes 210 ms
Question 5: how practical is maintaining a single and possibly distributed revocation authority?

Network log visualization (subset is shown, 18 transactions). Interval size is 20 ms. Experiment involves 5 users, 3 peers, 2 endorsements, 20 KiB/s and 50 KiB/s local and global bandwidths, and epoch length 5 seconds. Bars show objects in the network, lines show latencies (green for ideal, red for real). Latency scale is logarithmic.

- Despite short epochs (5 seconds), revocation requests do not result in any spikes in latency
- Overhead of issuing the non-revocation handle is 15 ms–30 ms — a fraction of total overhead
- We observed a stable 200 requests per second throughput on our testing machine
Question 6: what is the efficacy of the entire blockchain stack using our protocol?

- Number of endorsements does not significantly affect the overhead if endorsements are processed in parallel.
- Number of users influences the overhead substantially. Each user increases TXs validated by a single peer.
- Number of peers is positively correlated with the overhead.

TX is completed when the last peer validates it.
Anonymous Transactions with Revocation and Auditing in Hyperledger Fabric

Anonymous Credentials, Revocation, Auditing, Blockchain

DOI: 10.1007/978-3-030-92548-2_23

Dmytro Bogatov, Angelo De Caro, Kaoutar Elkhiyaoui, Björn Tackmann
dmytro@bu.edu, adc@zurich.ibm.com, kao@zurich.ibm.com
bjoern@dfinity.org

Built from bc167548 on December 25, 2021

Boston University
Graduate School of Arts and Sciences
Department of Computer Science
REFERENCES


