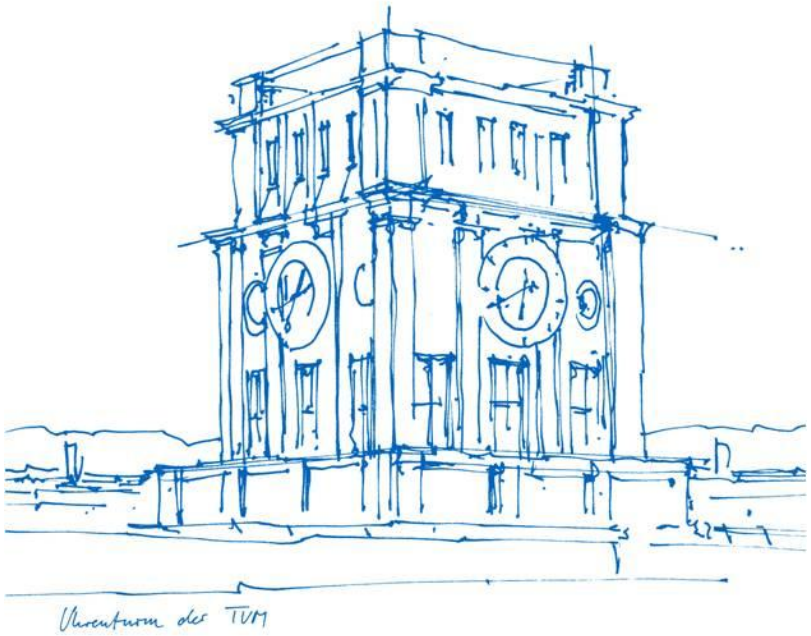
Advancing Privacy-Enhancing Technologies for Policy-Driven Data Sovereignty and Provenance

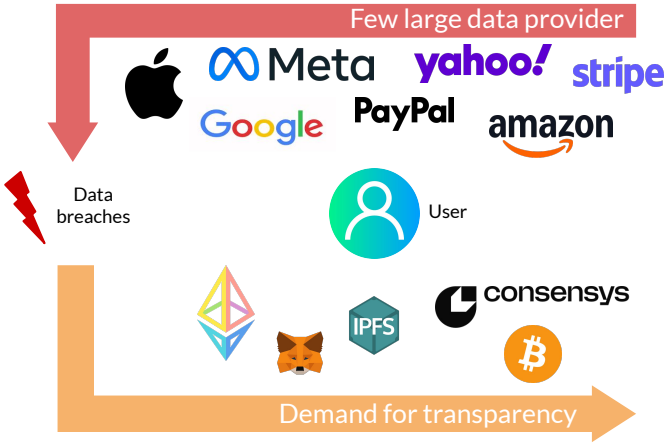


Jan Lauinger
Technical University of Munich
TUM School of Computation, Information and Technology
Associate Professorship of Embedded Systems and Internet of Things
Munich, 27. March 2025

From Trusted to Verifiable

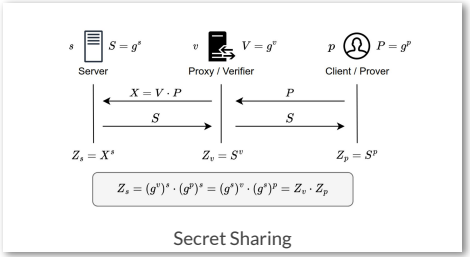
- Modern online services and devices face a data consent - protection dilemma
  - Digital user experience
  - Data silos at few “trusted” providers
  - Digital legislation on data sovereignty
  - Open public digital infrastructures

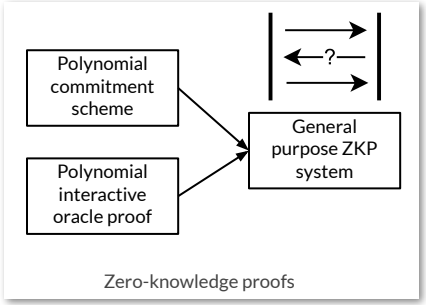
- Privacy-Enhancing Technologies (PETs) as the tool to redefine existing barriers

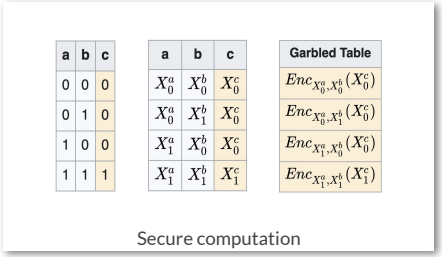


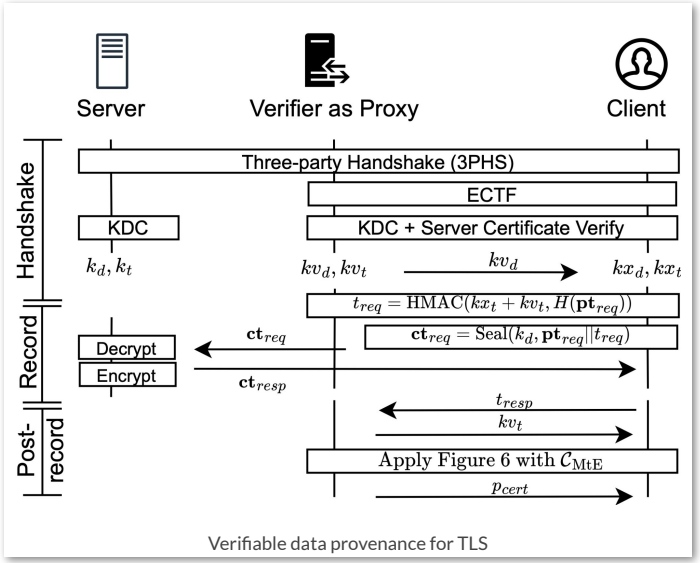
Research Question
How can we ensure self-determined protection and transparency of digital data during the creation and interaction of online accounts?

Researching Privacy-Enhancing Technologies









Structure of this work

Systematization of Data Sovereignty → Insights & Challenges → Contrib 1: Sovereign Identity & Data Privacy

- Alignment of sovereign principles in online protocols
- Access of public infrastructures to PETs
- New research standards are open for improvements

Portal, 7 pages, @**ICBC24**, c.1

A-PoA, 9 pages, @**ICBC21**, c.16

SoK, 21 pages @**EuroS&P23**, c.25

Systematization of Data Provenance → Insights & Challenges → Contrib 2: Policy-driven Data Provenance

- Limiting performance and communication requirements
- Gap between PETs and user experience

Janus, 20 pages, @**PETS25**, c.12

Artifacts a, f, r

Origo, 18 pages, @**PETS25**, c.8

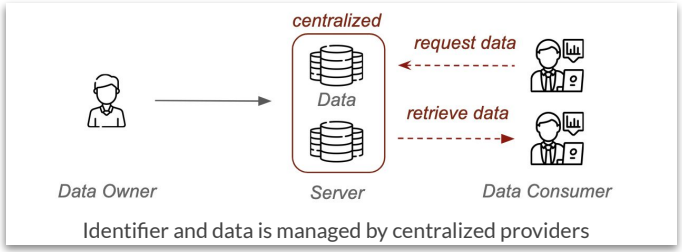
zkGen, 5 pages, @**ICBC24**, c.1

Contribution 1: Sovereign Identity & Data Privacy

Systematization of Data Sovereignty

Scope of digital data

- Identifier & Data Management
- Policy-driven Data Attestation
- Policy-driven Access Control
- Policy-driven Data Processing

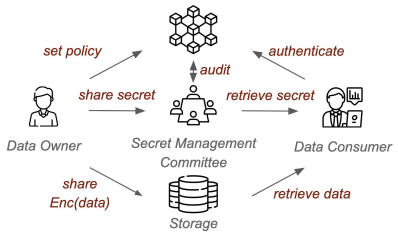


Identifier and data is managed by centralized providers

Methodology

- Formalized definitions of core functionalities in a unified system model
- SOTA/ Discussion to identify insights and challenges
- Assessment of academia/industry support of formalized functionalities

User-defined policy governs access to data



Publications

- SoK: Data Sovereignty. Published in: IEEE 8th European Symposium on Security and Privacy (EuroS&P), Delft, Netherlands, 03-07 July 2023. Pages: 21. Authors: J. Ernstberger, **J. Lauinger**, F. Elsheimy, L. Zhou, S. Steinhorst, R. Canetti, A. Miller, A. Gervais, D. Song

Contribution 1: Sovereign Identity & Data Privacy

Portal: Decentralizing Single Sign-On (SSO)

Challenge

Alignment of sovereign principles in online protocols.
Access of public infrastructures to PETs.

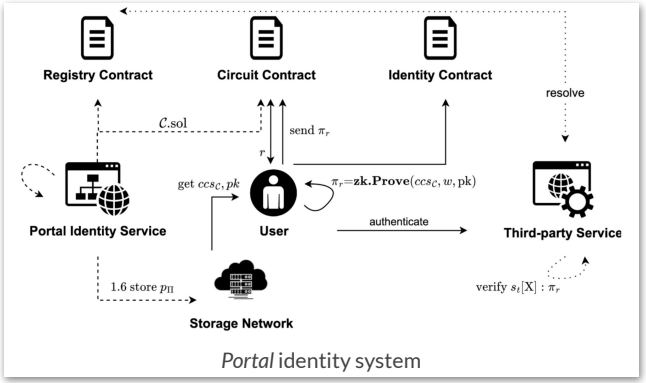
Solution

Time-bound and replay-resistant proofs on public blockchains as an core enabler for shifting data control and responsibility back to users.

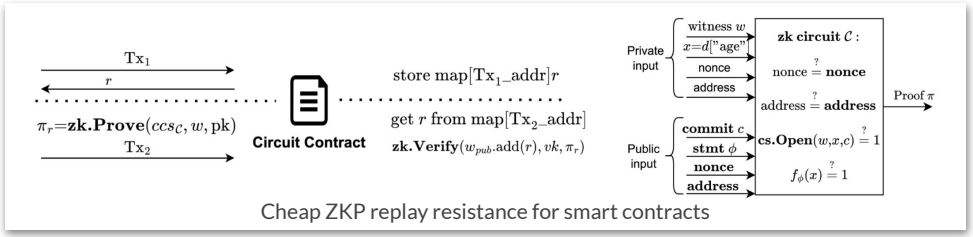
Contributions

Securing on-chain ZKP verifications against replay attacks.
Portal as an SSO alternative with enhanced privacy and control.
Open source proof of concept and cost evaluation.

User first online economy



Portal identity system



Cheap ZKP replay resistance for smart contracts

Publications

- Portal: Time-Bound and Replay-Resistant Zero-Knowledge Proofs for Single Sign-On. Published in: IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Dublin, May 2024. Pages: 7. Authors: **J. Lauinger**, S. Bezmez, J. Ernstberger, and S. Steinhorst

Contribution 1: Sovereign Identity & Data Privacy

A-PoA: Privacy-preserving Authorization for Digital Services

Challenge

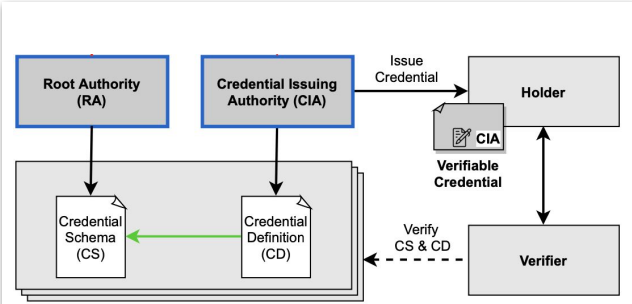
New research standards are open for improvements.

Solution

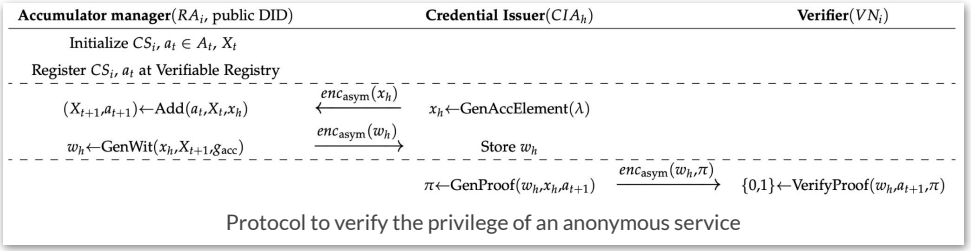
Providing privacy-enhanced permissions for online services.

Contributions

Protocol definition of *A-PoA* and open-source proof of concept.
Verifiable trust hierarchies among entities in certification systems.



Improving privacy in the verifiable credentials ecosystem



Protocol to verify the privilege of an anonymous service

Aligning consent and protection

Publications

- A-PoA: Anonymous Proof of Authorization for Dec. Identity Management. Published in: IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Australia, May 2021. Pages: 9. Authors: **J. Lauinger**, J. Ernstberger, E. Regnath, M. Hamad, S. Steinhorst

Contribution 1: Sovereign Identity & Data Privacy

Summary & Key Findings

Research question

How can we ensure self-determined protection and transparency of digital data during the creation and interaction of online accounts?

Answer

Understand which degree of self-determined protection can be build. **Translate** functionalities of existing online protocols into a context with strong sovereignty guarantees. **Apply** PETs to balance transparency and protection of data.

Future work

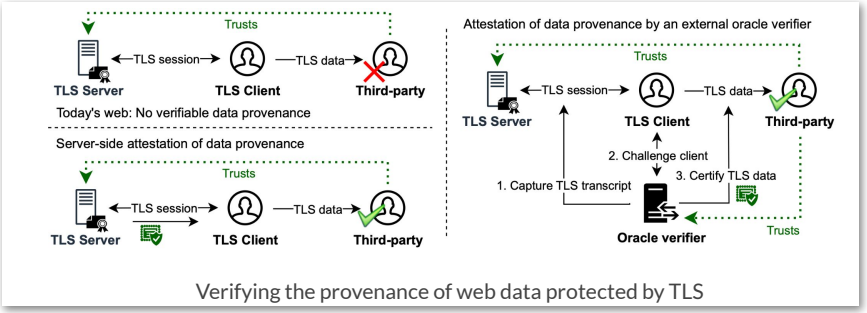
Technical development of control during mediated data administration (data access and computation on data).
Understand if new regulations, policies, and necessary functionalities demand a new degree of digital transparency and privacy.

Contribution 2: Policy-driven Data Provenance

Systematization of Data Provenance

Scope of data provenance

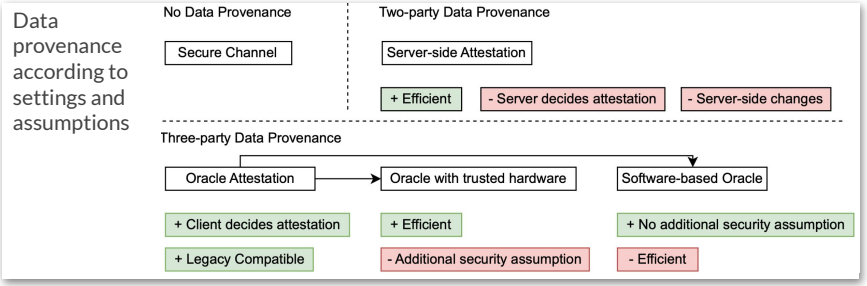
- Regain control over external data
- Policy-driven selection and presentation of data
- Trustworthy and verifiable origin of presented data



Verifying the provenance of web data protected by TLS

Methodology

- System settings
- Assumptions
- Deployment options



Contribution 2: Policy-driven Data Provenance

Origo: Bandwidth-optimized Data Provenance

Challenge

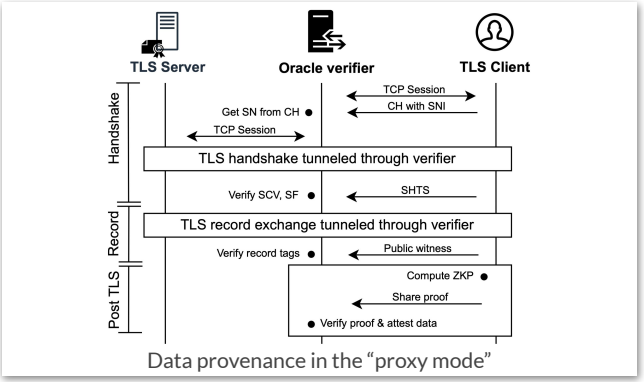
Limiting performance (ZKP computation efficiency). What is causing the limitation?

Solution

Consideration of data provenance in the proxy mode with the assumption that machine-in-the-middle attacks are infeasible.
Development of an optimized and custom ZKP circuit that connects commitments of TLS handshake parameters with record layer data.

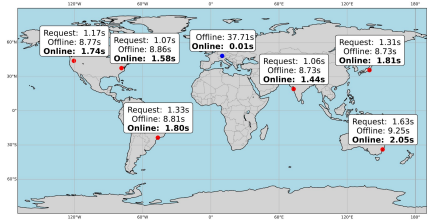
Contributions

New data provenance protocol for TLS 1.3 which outperforms state of the art in terms of bandwidth requirements. Open sourcing end-to-end code.



Data provenance in the "proxy mode"

Internet deployment



Adequately weak adversary unlocks high bandwidth savings

Publications

- Origo: Proving Provenance of Sensitive Data with Constant Communication.. Published in: The 25th Privacy Enhancing Technologies Symposium (PETS25), Washington DC, July 2025, Pages: 18. Authors: , J. Ernstberger*, **J. Lauinger***, Y. Wu, A. Gervais, and S. Steinhorst

Contribution 2: Policy-driven Data Provenance

Janus: Fast Privacy-preserving Data Provenance

Challenge

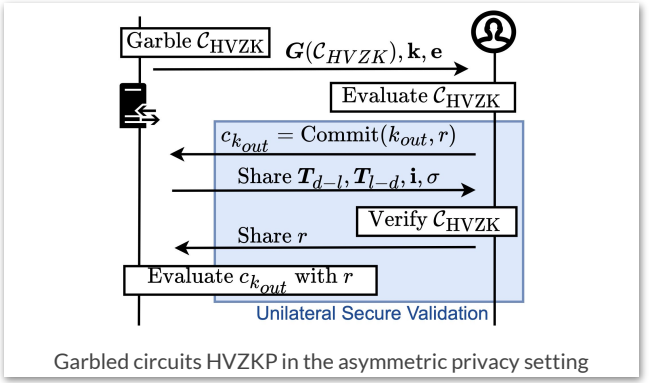
Limiting performance (ZKP computation efficiency). What is causing the issue?

Solution

Improving ZKP computations using a 1-RTT protocol extension to secure an HVZK proof system (in the asymmetric privacy setting) against malicious adversaries.

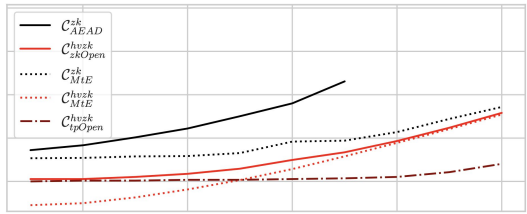
Contributions

Formalizing the asymmetric privacy setting using a secret sharing scheme and a cryptographic commitment scheme. Efficiency improvements for data provenance in TLS 1.2 and TLS 1.3. Open sourcing code.



Garbled circuits HVZKP in the asymmetric privacy setting

Scalability analysis



Less security assumptions at low cost

Publications

- Janus: Fast Privacy-Preserving Data Provenance For TLS. Published in: The 25th Privacy Enhancing Technologies Symposium (PETS25), Washington DC, USA, 14–19 July 2025. Pages 20. Authors: **J. Lauinger**, J. Ernstberger, A. Finkenzeller, and S. Steinhorst

Contribution 2: Policy-driven Data Provenance

zkGen: Policy-driven Creation of Private Computations

Challenge

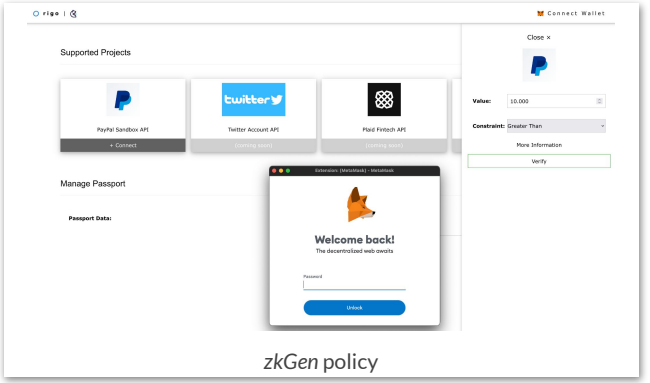
Provision of fixed circuits causes limited user experience.

Solution

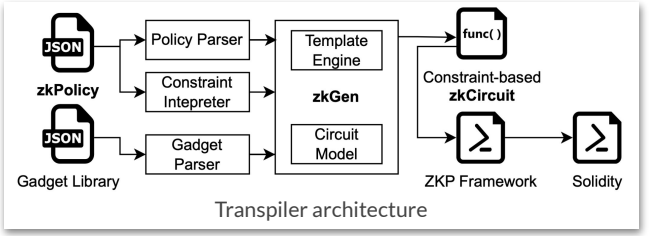
Transpiling human readable policies (created by the user) into private computation circuits.

Contributions

Transpiler architecture and new JSON policy language.
Automated generation of user-driven PET circuits.



zkGen policy



Transpiler architecture

Circuits for dynamic data needs

Publications

- zkGen: Policy-to-Circuit Transpiler. Published in: IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Dublin, Ireland, 27-31 May 2024. Pages: 5. Authors: **J. Lauinger**, J. Ernstberger, and S. Steinhorst

Contribution 2: Policy-driven Data Provenance

Summary & Key Findings

Research questions

How can we practically return the sovereignty of custodial data back to devices and users?
How can we mitigate unconscious sharing and processing of custodial data through user-driven policies?

Answer

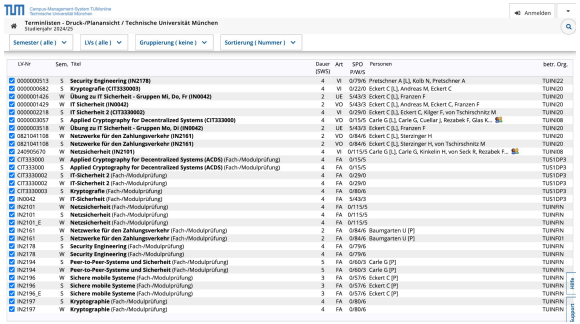
The tailored interplay between security assumptions, adapted PETs, and protocol-specific conditions leads to substantial improvements.
Integrate missing components (*zkGen*) to data provenance protocols to enable a policy-driven verification of user data..

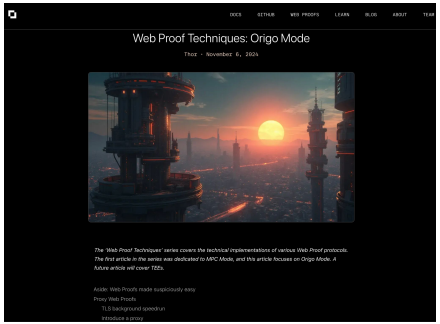
Future work

Integrating & measuring data provenance protocols in production environments (browsers).
Commitment attack feasibility analysis is an interesting direction to continue to lower security requirements (e.g. remove key derivation circuits in *Origo*).
Investigate different deployment types to evaluate censorship resistance.

Conclusion

Summary & Impact





What does this mean?

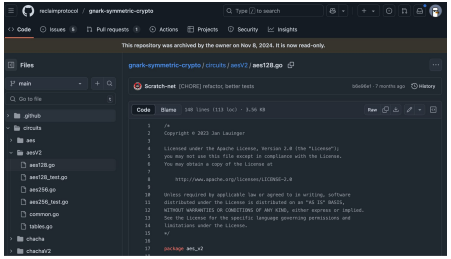
You **can** adapt Internet protocols by integrating PETs and this leads to a better balance between consent & protection of user data.

Is our research applied somewhere?

Pluto.xyz builds their data provenance solution using the *Origo* protocol.
Reclaimprotocol.org builds upon our gnark AES zk circuit.
Talks @TUM Blockchain Salon 23/24/25, @TUM Blockchain Conference 24, @Blockchain Knowledge Exchange (new TUM course "Applied Cryptography for Decentralized Systems lecture" CIT333000)

Future directions

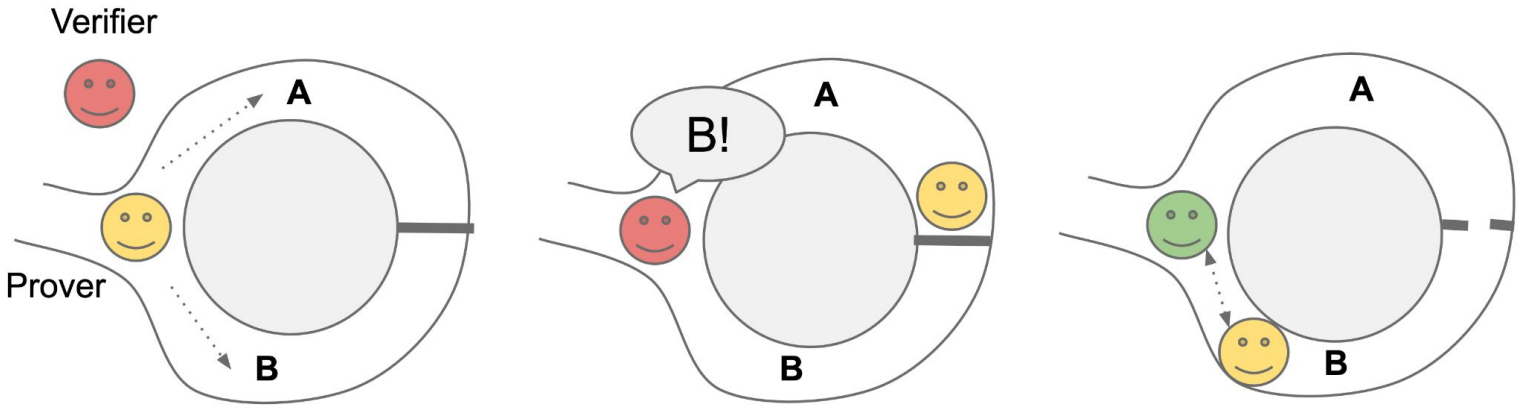
Continue research and publish systematization of data provenance + ZKPs in the browser.
Educate industry about new capabilities enabled by PETs.
Advertise and continue to optimize our approaches that enhance data sovereignty.



Backup slides

Backup: Background

Zero-knowledge Proofs (ZKPs) - intuition



Backup: Background

ZKPs - properties

- **Completeness**
  - Prover *P*, **knowing** a solution to a problem, can successfully convince a verifier *V*.
- **Soundness**
  - Prover *P*, **not** knowing a solution to a problem, will fail to convince a verifier *V*.
- **Zero Knowledge**
  - A **zero knowledge** Proof of Knowledge (PoK) scheme requires the verifier *V* to learn nothing but the validity of a convincing assertion from prover *P*.

Backup: Background

ZKPs - dedicated

Prover		Verifier
$\alpha \in \mathbb{Z}_q$		$(U, A = U^\alpha) \in \mathbb{G}$
$r \leftarrow \mathbb{Z}_q$	$\xrightarrow{R = g^r \in \mathbb{G}}$	
	$\xleftarrow{c \in C}$	$c \leftarrow C = \{1, \ldots, 2^\lambda\} \subseteq \mathbb{Z}_q$
$z = c \cdot \alpha + r \in \mathbb{Z}$	$\xrightarrow{z}$	accept iff $U^z = A^c \cdot R$

Figure: The Schnorr Proof of Knowledge protocol [2], as shown in [3].

1. Completeness Commitment scheme (opening, challenge, response)

$U^z = g^z = g^{c \cdot \alpha + r} = g^r \cdot (g^\alpha)^c = R \cdot (U^\alpha)^c = R \cdot A^c$

2. Soundness *Extractor* concept with the ability to extract secret knowledge from P convinces V of the existence of a satisfying solution.

2. Soundness Assumption: V is able to receive two accepting conversations (R, c, z) and (R, c', z').

With $U^z = A^c \cdot R$ and $U^{z'} = A^{c'} \cdot R$

$\implies \quad U^{z - z'} = A^{c - c'}$

$\implies \quad \alpha = \frac{z - z'}{c - c'} = \log_g(U)$

3. HVZK *Simulator* concept with simulated transcript T_{sim} and real transcript T_{real} of interactive protocol [4].

3. Honest Verifier Zero Knowledge

Select $z, c \leftarrow \mathbb{Z}_q$

Calculate $\alpha = \frac{g^z}{U^c}$

Output $T_{sim} = (\alpha, c, z)$

4. Zero Knowledge (Fiat-Shamir Heuristic)

$c = H(R) \implies c = H(T)$

Backup: Background

ZKPs - advanced

GenProof (w_x, x, a_t) :

$k, \rho_x, \rho_k \xleftarrow{R} [-B, B]$;
$A_g = g^k h^{\rho_k}$;
$l \leftarrow H_{\text{prime}}(w_x, a_t, z, A_g, A_{w_x})$;
$q_x \leftarrow \lfloor (k + c \cdot x)/l \rfloor$;
$r_x \leftarrow (k + c \cdot x) \mod l$;
$\pi \leftarrow \{l, z, g^{q_x} h^{q_\rho}, w_x^{q_x}, r_x, r_\rho\}$
return : π

$z = g^x h^{\rho_x}$
$A_{w_x} = w_x^k$
$c \leftarrow H(l)$
$q_\rho \leftarrow \lfloor (\rho_k + c \cdot \rho_x)/l \rfloor$
$r_\rho \leftarrow (\rho_k + c \cdot \rho_x) \mod l$

VerifyProof (w_x, a_t, π) :

$\{l, z, Q_g, Q_{w_x}, r_x, r_\rho\} \leftarrow \pi$;
$A_g \leftarrow Q_g^l g^{r_x} h^{r_\rho} z^{-c}$;
Verify $r_x, r_\rho \in [l]$;
return : $\{0, 1\}$

$c = H(l)$
$A_w \leftarrow Q_{w_x}^l w_x^{r_x} a_t^{-c}$
$l = H_{\text{prime}}(w_x, a_t, z, A_g, A_w)$

Completeness

Accept if: $\Rightarrow Q_g^l \cdot g^{r_x} \cdot h^{r_\rho} = (g^{q_x} \cdot h^{q_\rho})^l \cdot g^{r_x} \cdot h^{r_\rho} = g^{q_x \cdot l + r_x} \cdot h^{q_\rho \cdot l + r_\rho} = g^{s_x} \cdot h^{s_\rho} = g^{k + c \cdot x} \cdot h^{\rho_k + c \cdot \rho_x} = g^k \cdot h^{\rho_k} \cdot g^{x \cdot c} \cdot h^{\rho_x \cdot c} = A_g \cdot z^c$
$\Rightarrow Q_u^l \cdot u^{r_x} = (u^{q_x})^l \cdot u^{r_x} = u^{q_x \cdot l + r_x} = u^{s_x} = u^{k + c \cdot x} = u^k \cdot u^{x \cdot c} = A_u \cdot w^c$

Soundness (Extractor)

1. Extractors to extract x, ρ such that $z = g^x \cdot h^\rho$ and $g^{s_1} \cdot h^{s_2} = A_g \cdot z^c$.
2. Set $R \leftarrow \{\}$ and sample $s_1, s_2 \xleftarrow{R} [0, 2^\lambda]$.
3. Sample $l \xleftarrow{R} \text{Primes}(\lambda)$, $c \xleftarrow{R} [0, 2^\lambda]$ and send s_1, s_2, l, c to $\mathscr{A}_1$.
4. Obtain output Q_g, Q_u, r_1, r_2 from $\mathscr{A}_0$. If transcript is accepting ($Q_g^l \cdot g^{r_1} \cdot h^{r_2} = A_g \cdot z^c$ and $Q_u^l \cdot u^{r_1} = A_u \cdot w^c$) then update $R \leftarrow R \cup \{(r_1, r_2, l, c)\}$. Otherwise return to step 2.
5. Use CRT to compute $s_1 = r_1^{(i)} \mod l^{(i)}$ and $s_2 = r_2^{(i)} \mod l^{(i)}$, for each $(r_1^{(i)}, r_2^{(i)}, l^{(i)}) \in R$. If $u^{s_1} = A_u \cdot w^c$ then output s_1, otherwise return to step 2.
6. Repeat for s_1', s_2', c', so that $x = \Delta s_1 / \Delta c = (s_1 - s_1')/(c - c')$ and $\rho = \Delta s_2 / \Delta c = (s_2 - s_2')/(c - c')$, with extraction based on $u^{s_1} = A_u \cdot w^c$ and $u^{s_1'} = A_u \cdot w^{c'}$, thus $(u^x)^{\Delta c} = w^{\Delta c} \Rightarrow u^x = w$.

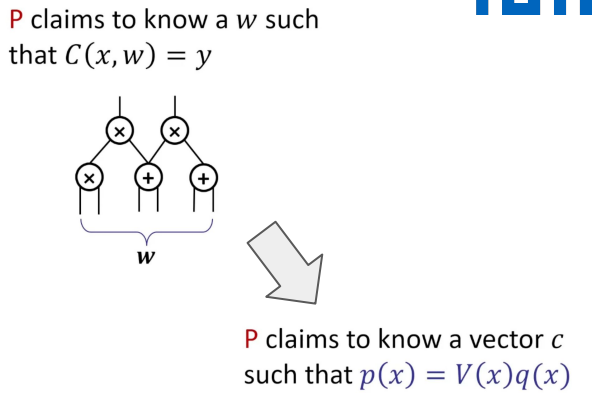
Zero Knowledge (Simulator)

1. $\widetilde{c} \xleftarrow{R} [0, 2^\lambda]$, $\widetilde{l} \xleftarrow{R} \text{Primes}(\lambda)$
2. $\widetilde{z} \leftarrow h^{\widetilde{\rho}}$, $\rho \xleftarrow{R} [B]$
3. $\widetilde{q}_x, \widetilde{q}_r \xleftarrow{R} [B]^2$
4. $\widetilde{r}_x, \widetilde{r}_\rho \in [l]^2$
5. $\widetilde{Q}_g \leftarrow g^{\widetilde{q}_x} \cdot h^{\widetilde{q}_\rho}$, $\widetilde{Q}_u \leftarrow u^{\widetilde{q}_x}$
6. $\widetilde{A}_g \leftarrow \widetilde{Q}_g^l \cdot g^{\widetilde{r}_x} \cdot h^{\widetilde{r}_\rho} \cdot z^{-\widetilde{c}}$, $\widetilde{A}_u \leftarrow \widetilde{Q}_u^l \cdot u^{\widetilde{r}_x} \cdot w^{-\widetilde{c}}$

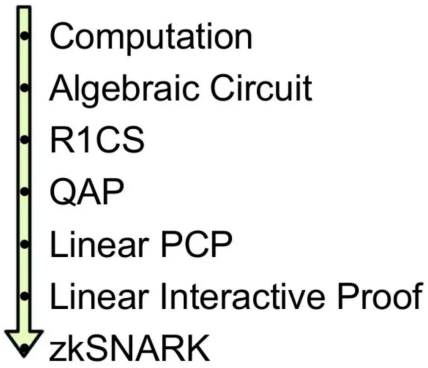
$\Rightarrow (\widetilde{z}, \widetilde{A}_g, \widetilde{A}_u, \widetilde{c}, \widetilde{l}, \widetilde{Q}_g, \widetilde{Q}_w, \widetilde{r}_x, \widetilde{r}_\rho)$ is statistically indistinguishable from $(z, A_g, A_u, c, l, Q_g, Q_w, r_x, r_\rho)$

Backup: Background

ZKPs - general purpose proof systems

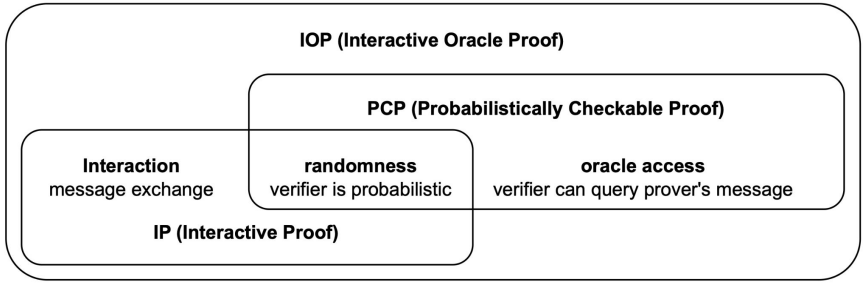


- Arithmetic representations & circuit satisfiability
  - R1CS, Plonkish, AIR, Custom CSS
- Quadratic arithmetic program - QAP (System of equations over polynomials)
- Functional commitment scheme (cryptographic object)
  - Cryptographic setup procedure?
- Compatible interactive oracle proof - IOP (information theoretic object)
  - Polynomial, Multilinear, Vector, etc.. IOP



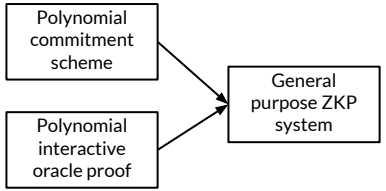
Backup: Background

ZKPs - types



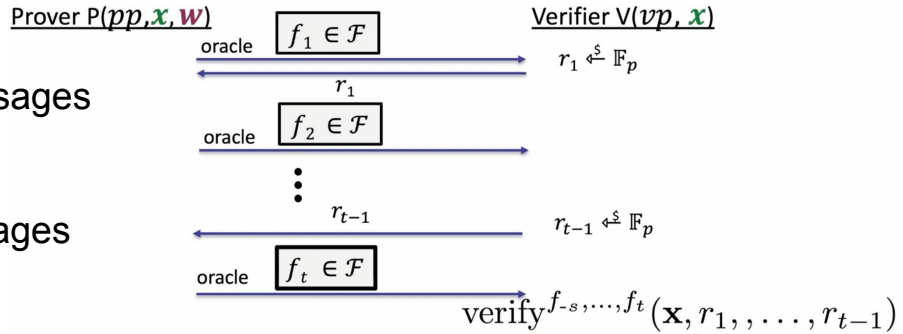
Popular commitment schemes

- Polynomial, Multilinear, Vector (Merkle tree) commitment
- Inner product commitments (inner product arguments - IPAs)



Verifier access (powers of the verifer) in IOPs

- Oracle access: verifier may query provers messages
- Randomness: verifier is probabilistic
- Interaction: prover and verifier exchange messages
- Multi-prover: multiple provers that are isolated



Backup: Background

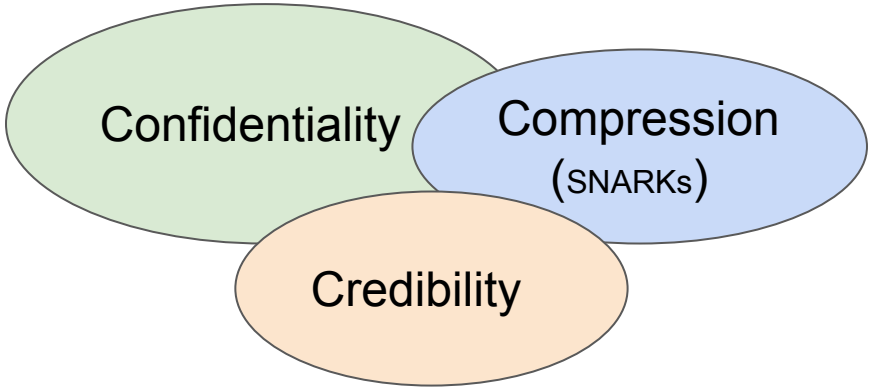
ZKPs - commitments and types of proof systems

- Bilinear Groups → KZG commitment
- Hash functions → e.g. for Fast Reed-Solomon IOP of Proximity- FRI
- Elliptic curves → for Bulletproofs
- Groups of unknown order

- zkSNARK → Succinctness: short proofs and fast to verify
- zkSTARK → Scalable, transparent
- MPC-based ZKP → Efficient evaluation of non-algebraic statements
- Recursive ZKPs → Proof of proof
- zkEVMs → Efficient ZK computation for any type of computation

Backup: Background

ZKPs - use cases

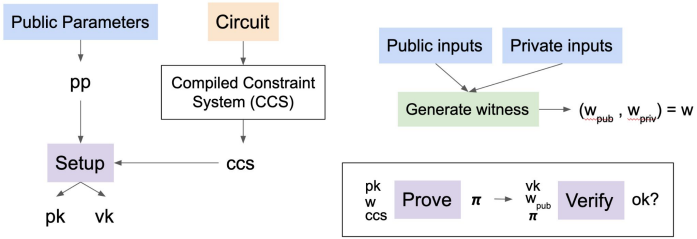


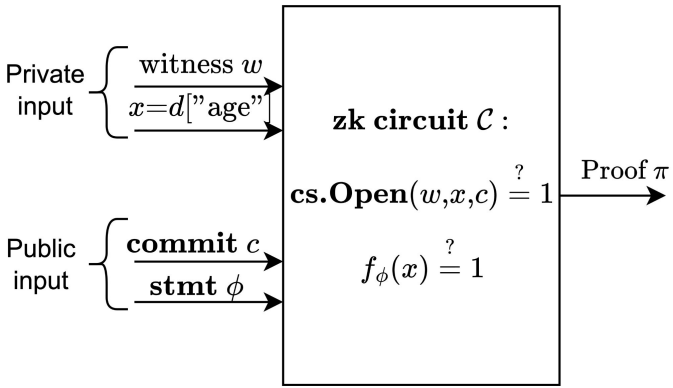
- **Privacy** → Credentials, group membership
- **Compression** → Rollups, compact credentials
- **Content Provenance** → zkTLS, image compression, data feeds, zkBridges
- **Blockchain Applications** → Tornado cash, Zcash

Backup: Background

ZKPs - in practice

- **ganache -m "much repair shock … bullet interest solution"**
  - Starting local blockchain network (single instance)
- **go run main.go -init true**
  - Compiles a gnark circuit (cubic function)
  - Runs a *ZKP* setup procedure to generate prover and verifier keys
  - Exports a solidity proof verification contract
- **go run main.go -bindings true**
  - Compile contract into ABI and BIN files with *solc*
  - Use *abigen* to generate Go code bindings in a Go *verifier* package
- **go run main.go -deploy true**
  - Use Go *verifier* package to deploy the contract and interact with it
- **go run main.go -address 0x2e144…c888f1fF1a -verify true**
  - Call the contract to verify the computed proof





Backup: Background

Secure Two-party Computation (2PC)

Collaboratively compute a function and maintain input secrecy

- **Multi-party Computation (MPC)**: MPC is the joint computation of a public function f(**x**)=y between M parties with input x_M such that no party learns the private inputs of the counterparties
  - The **adversary** is assumed to corrupt T parties.
- **Secure two-party computation (2PC)**: Secure 2PC uses M=2 and T=1 such that two parties with private inputs x_1, x_2 can collaboratively compute f (x_1, x_2) without learning any other x_i

Backup: Background

2PC - adversary model

Assumption

- **Semi-honest parties** honestly follow the protocol specification
  - Attack/Try to learn from exchanged parameters
- **Malicious adversaries** arbitrarily deviate from the protocol specification
  - Inject false values such that the opponent accepts values without notice
  - Selective-failure attack: Inject false values, then observe and learn from failure
    - (e.g. know secret permutation & corrupt a row, learn which row was evaluated, learn information on inputs)

Backup: Background

2PC - types

Constructions

- **Garbled circuits**
  - Based on oblivious transfer (OT), OT cost depends on input size
  - Two-party computation
  - Constant communication (independent of circuit depth)

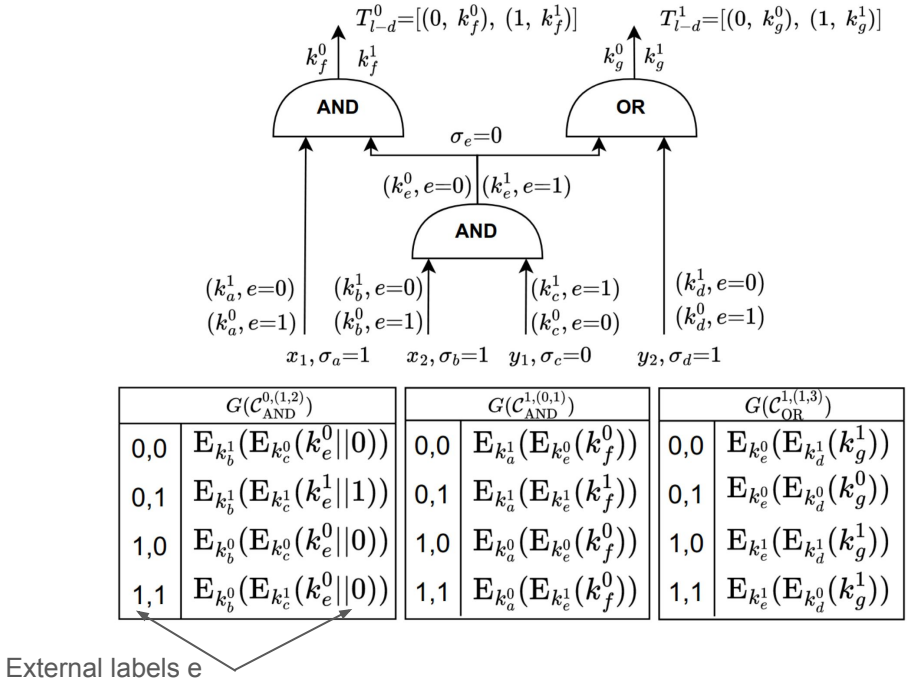
- **GMW, BGW, CCD**
  - Based on secret sharing
  - Multi-party computation
  - Boolean (AND, OR) or arithmetic (MUL, ADD) circuits
  - Gate-by-gate computation in multiple oblivious communication rounds for MUL operation

Backup: Background

2PC - example

Garbled circuit parameters

- **Parties**: Garbler and evaluator
- **Boolean circuit**: AND, OR gates
- **Wire labels**: k, i, e, sigma
- **Private labels**: i (internal), sigma
- **Public labels**: e (external), k^i
- **Random labels**: sigma, k^i



Backup: Background

2PC - example

Example: Garbler input **x**=[x_1=1,x_2=0], Evaluator input **y**=[y_1=0,y_2=1], σ_a=1,σ_b=1,σ_c=0,σ_d=1

1. **Garbler garbles**: sample sigmas, k_L^i with L ∈ {a,b,c,d,e,f,g} (16B with aes128), compute e_L = sigma xor i, compute G tables, send (**G,** circuit C, (k^1_a, e=0), (k^0_b, e=1), $T_{l\text{-}d}$)

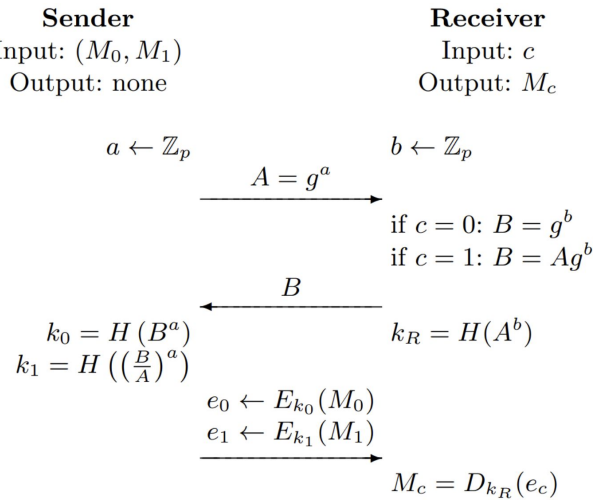
2. **Garbler & Evaluator using OT**: for every input wire corresponding to y input bit, run
   1-out-of-2 OT with (m_1=[k^0_L, e], m_2=[k^1_L, e]).
   $OT_{c,d}$ with b_c=0, b_d=1 yields (k^0_c, e=0), (k^1_d, e=0)

Backup: Background

2PC - example

Transfer 2 messages without learning which message is picked

- 1-out-of-2 OT with 2 messages k_0, k_1
- Evaluator obtains (k_b ,e), with b ∈ {0,1}



Backup: Background

2PC - example

Example: Garbler input **x**=[x_1=1,x_2=0], Evaluator input **y**=[y_1=0,y_2=1], σ_a=1,σ_b=1,σ_c=0,σ_d=1

2. **Garbler & Evaluator using OT**: for every input wire corresponding to y input bit, run
   1-out-of-2 OT with (m_1=[k^0_L, e], m_2=[k^1_L, e]).
   $OT_{c,d}$ with b_c=0, b_d=1 yields (k^0_c, e=0), (k^1_d, e=0)

3. **Evaluator use G tables to evaluate**
   Use G_{AND0} and (k^0_b, e=1), (k^0_c, e=0) -> row (1,0) to obtain (k^0_e, e=0)
   Use G_{AND1} and (k^1_a, e=0), (k^0_e, e=0) -> row (0,0) to obtain (k^0_f)
   Use G_{OR1} and (k^0_e, e=0), (k^1_d, e=0) -> row (0,0) to obtain (k^1_g)
   Use $T_{l\text{-}d}$ map to obtain out_0=0 from k^0_f and out_1=1 from k^1_g

Backup: Background

2PC - example

Example: Garbler input **x**=[x_1=1,x_2=0], Evaluator input **y**=[y_1=0,y_2=1], σ_a=1,σ_b=1,σ_c=0,σ_d=1

3. **Evaluator use G tables to evaluate**

   …

   Use $T_{l\text{-}d}$ map to obtain out_0=0 from k^0_f and out_1=1 from k^1_g

4. **Share output back to garbler**

   With (out_0=0, out_1=1) garbler knows wire keys at output labels

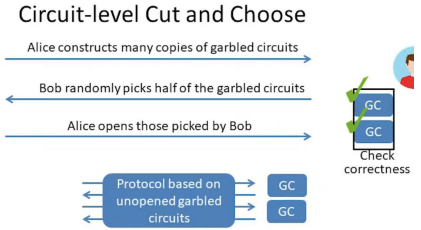
   However, ambiguity of wire keys in G_{AND1} together with OT obfuscates input of evaluator

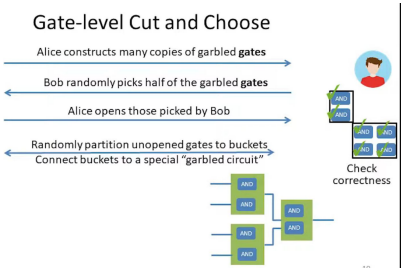
Backup: Background

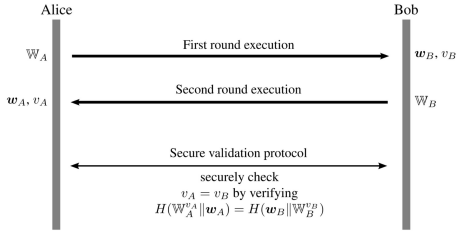
2PC - maliciously secure 2PC

Techniques to secure 2PC against malicious adversaries

- **Cut-and-choose**: many copies of garbled circuits, validate random subset, use unopened circuits
  - Exist at a circuit and gate level
- **Dual execution**: two rounds semi-honest 2pc + secure validation
- **Authenticated garbling**: malicious secret sharing of GC permutation bits
  - Based on TinyOT protocol

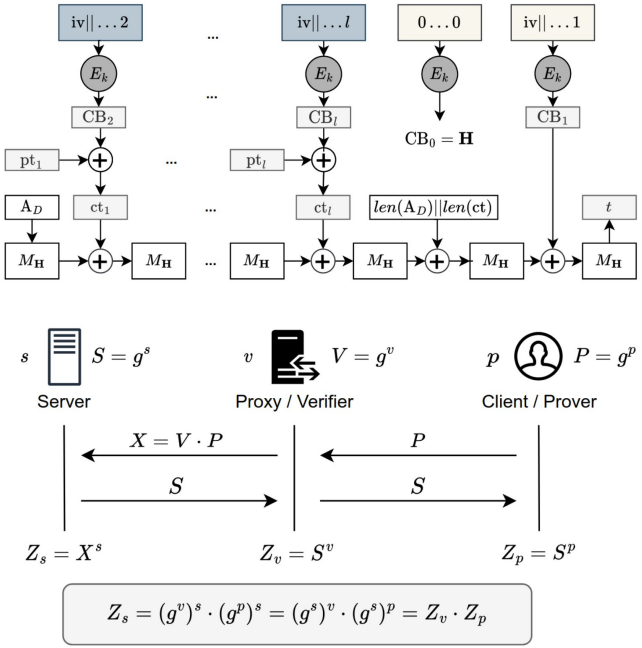






Backup: Background

Transport Layer Security (TLS)



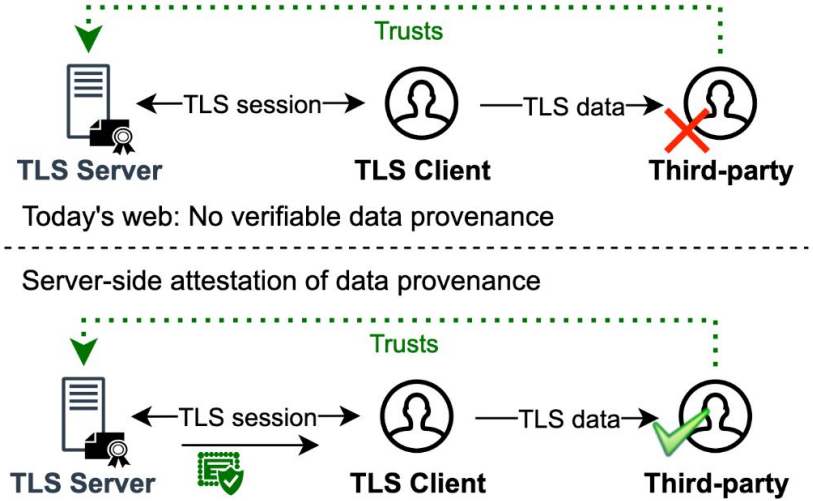
TLS Handshake between the client c and server s:

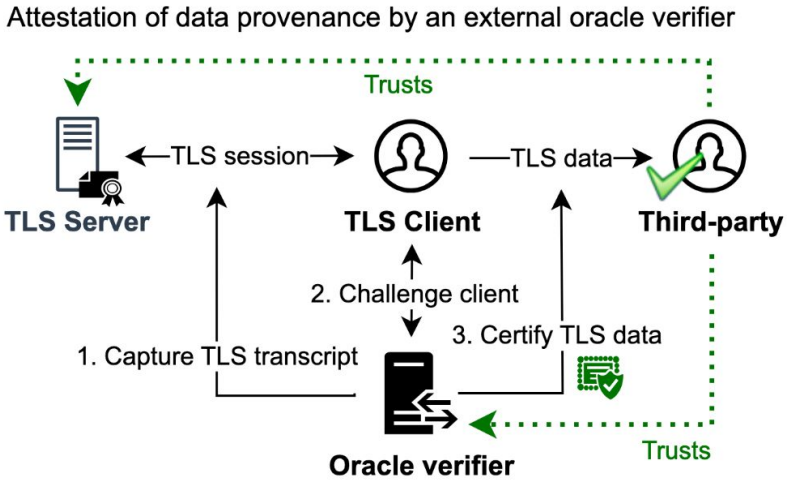
inputs: $x \xleftarrow{\$} \mathbb{F}_p$ by c. $(y \xleftarrow{\$} \mathbb{F}_p, sk_S, pk_S)$ by s.
outputs: $(\text{tk}_{\text{CATS}}, \text{iv}_{\text{CATS}}, \text{tk}_{\text{SATS}}, \text{iv}_{\text{SATS}})$ to c and s.

1. c: $X = x \cdot G$; send X in m_{CH}
2. s: $Y = y \cdot G$; send Y in m_{SH}
3. b: dES = **hkdf.exp**(**hkdf.ext**(0,0),"derived" || H(""))
4. b: DHE = $x \cdot y \cdot G$; HS = **hkdf.ext**(dES, DHE)
5. b: SHTS = **hkdf.exp**(HS,"s hs traffic" || H_2)
6. b: CHTS = **hkdf.exp**(HS,"c hs traffic" || H_2)
7. b: $(\text{k}_{\text{CHTS}}, \text{iv}_{\text{CHTS}})$ = DeriveTK(CHTS)
8. b: $(\text{k}_{\text{SHTS}}, \text{iv}_{\text{SHTS}})$ = DeriveTK(SHTS)
9. b: fk_S = **hkdf.exp**(SHTS, "finished" || "")
10. s: SCV=**ds.Sign**(sk_S,label_{11}||H_6); send SCV in m_{SCV}
11. s: SF = **hmac**(fk_S, H_7); send SF in m_{SF}
12. c: SF' = **hmac**(fk_S, H_7); verify SF' $\stackrel{?}{=}$ SF
13. c: **ds.Verify**(pk_S, label_{11} || H_6, SCV) $\stackrel{?}{=}$ 1
14. b: fk_C = **hkdf.exp**(CHTS, "finished" || "")
15. c: CF = **hmac**(fk_C, H_9); send CF in m_{CF}
16. s: CF' = **hmac**(fk_C, H_9); verify CF' $\stackrel{?}{=}$ CF
17. b: dHS = **hkdf.exp**(HS,"derived" || H(""))
18. b: MS = **hkdf.ext**(dHS, 0)
19. b: CATS = **hkdf.exp**(MS, "c ap traffic" || H_3)
20. b: SATS = **hkdf.exp**(MS, "s ap traffic" || H_3)
21. b: $(\text{k}_{\text{CATS}}, \text{iv}_{\text{CATS}})$ = DeriveTK(CATS)
22. b: $(\text{k}_{\text{SATS}}, \text{iv}_{\text{SATS}})$ = DeriveTK(SATS)

Backup: Background

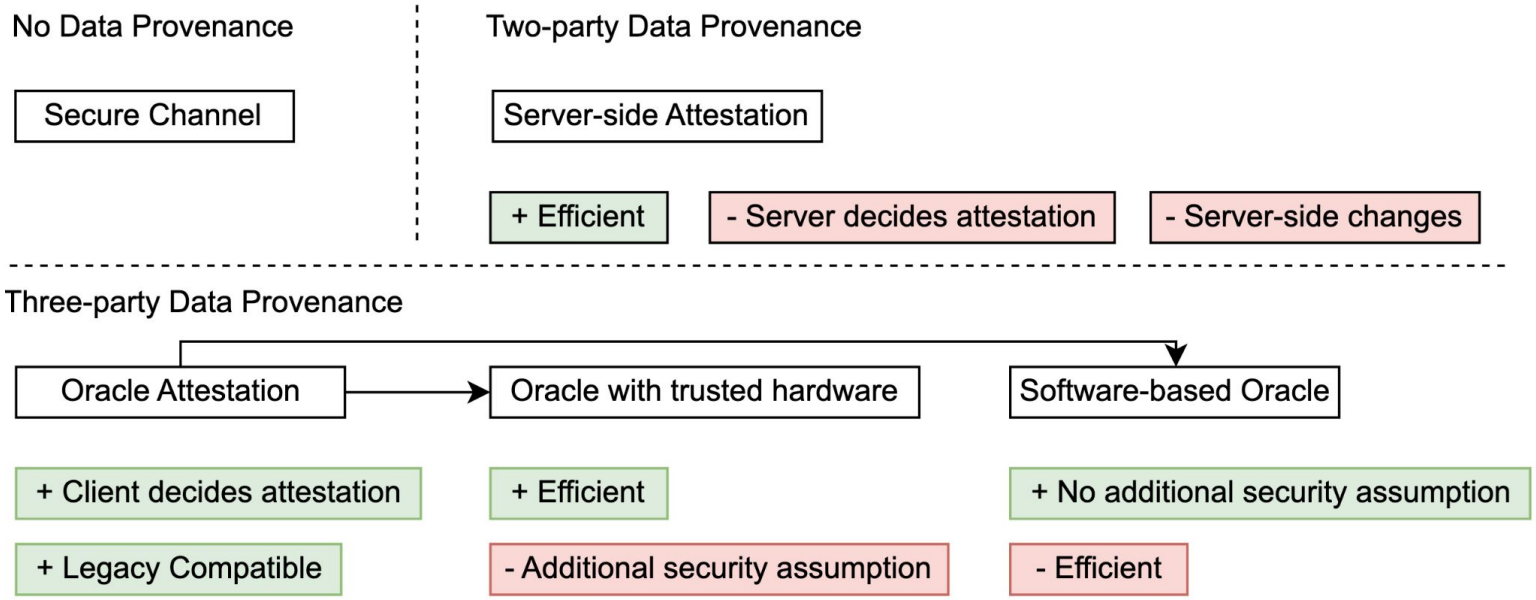
Data provenance





Backup: Background

Data provenance



Backup: Background

Data provenance requires more computation

ECTF between two parties p_1 and p_2.

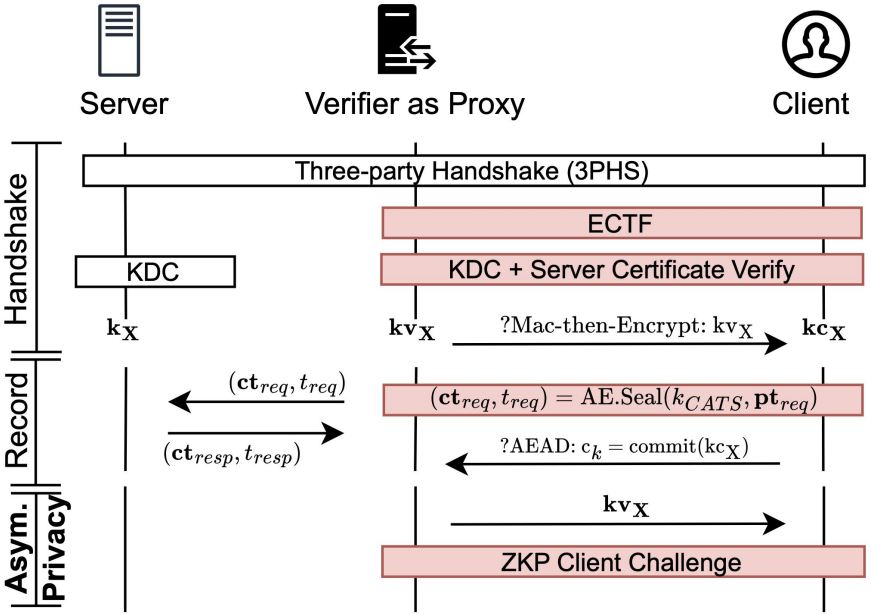
inputs: $P_1 = (x_1, y_1)$ by p_1, $P_2 = (x_2, y_2)$ by p_2.
outputs: s_1 to p_1, s_2 to p_2.

p_1: (sk,pk)=**mta.Setup**(1^λ); send pk to p_2
p_1: $\rho_1 \xleftarrow{\$} \mathbb{Z}_p$; **c1**=**mta.Enc**$([-x_1,\rho_1], sk)$; send **c1** to p_2
p_2: $\rho_2 \xleftarrow{\$} \mathbb{Z}_p$;(c2,$\beta$)=**mta.Eval**(**c1**,$[\rho_2,x_2]$,$pk$);$\delta_2 = x_2 \cdot \rho_2 + \beta$; send (c2,$\delta_2$) to p_1
p_1: α=**mta.Dec**(c2,sk);$\delta_1 = -x_1 \cdot \rho_1 + \alpha$;$\delta = \delta_1 + \delta_2$;$\eta_1 = \rho_1 \cdot \delta^{-1}$; **c1**=**mta.Enc**$([-y_1,\eta_1], sk)$; send (**c1**,$\delta_1$) to p_2
p_2: $\delta = \delta_1 + \delta_2$; $\eta_2 = \rho_2 \cdot \delta^{-1}$; (c2, β)=**mta.Eval**(**c1**,$[\eta_2,y_2]$,pk); $\lambda_2 = y_2 \cdot \eta_2 + \beta$; send c2 to p_1
p_1: α=**mta.Dec**(c2,sk); $\lambda_1 = -y_1 \cdot \eta_1 + \alpha$; **c1**=**mta.Enc**$([\lambda_1], sk)$; send **c1** to p_2
p_2: (c2,β)=**mta.Eval**(**c1**, $[\lambda_2]$, pk); $s_2 = 2 \cdot \beta + \lambda_2^2 - x_2$; send c2 to p_1
p_1: α=**mta.Dec**(c2,sk); $s_1 = 2 \cdot \alpha + \lambda_1^2 - x_1$

- **mta.Setup**$(1^\lambda) \rightarrow (sk_P,pk_P)$ takes in the security parameter λ and outputs a Paillier key pair (sk_P,pk_P).
- **mta.Enc**(**x**,sk_P) $\rightarrow$ (**c1**) takes in a vector of field elements **x**=$[x_1, \ldots, x_l]$ and a private key sk_P and outputs a vector of ciphertexts **c1**=$[E_{sk_P}(x_1), \ldots, E_{sk_P}(x_l)]$.
- **mta.Eval**(**c1**,**y**,pk_P) $\rightarrow$ (c_2,β) takes in the vector of ciphertexts **c1**=$[\mathrm{c1}_1, \ldots, \mathrm{c1}_l]$, a vector of field elements **y**=$[y_1, \ldots, y_l]$, and a public key pk_P. The output is a tuple of a ciphertext $\mathrm{c2} = \mathrm{c1}_1^{y_1} \cdot \ldots \cdot \mathrm{c1}_l^{y_l} \cdot E_{pk_P}(\beta')$ and the share $\beta = -\beta'$, where $\beta' \xleftarrow{\$} \mathbb{Z}_p$.
- **mta.Dec**(c2,sk_P) $\rightarrow (\alpha)$ takes as input a ciphertext c2 and a private key sk_P and outputs the share $\alpha = D_{sk_P}(\mathrm{c2})$.

Backup: Background

Data provenance requires more computation



$\mathcal{C}_{\text{MtE}}(\mathbf{pt}, kc, kc^t\ ;\ kv, kv^t, \mathbf{ct}_\alpha{=}\mathbf{ct}[\text{end-3:}], \phi)$:

1. $t' = \text{HMAC}(kc^t + kv^t, \mathbf{pt})$
2. $\mathbf{ct}_\alpha' = \text{AES}(kc + kv, t')$
3. assert: $\mathbf{ct}_\alpha \stackrel{?}{=} \mathbf{ct}_\alpha'$, $1 \stackrel{?}{=} f_\phi(\mathbf{pt})$

$\mathcal{C}_{\text{AEAD}}(\mathbf{pt}, kc\ ;\ kv, \mathbf{ct}, \text{iv}, t_\alpha, c_k, \phi)$:

1. $t_\alpha' = [\ \text{AES}(kc + kv, \mathbf{0})\ , \text{AES}(kc + kv, \text{iv}\,||\,\mathbf{1})\ ]$
2. $\mathbf{ct}' = \text{AES}(kc + kv, \mathbf{pt})$; $c_k' = \text{commit}(kc)$
3. assert: $t_\alpha \stackrel{?}{=} t_\alpha'$, $c_k \stackrel{?}{=} c_k'$, $\mathbf{ct} \stackrel{?}{=} \mathbf{ct}'$, $1 \stackrel{?}{=} f_\phi(\mathbf{pt})$

Backup: Policy-driven Data Provenance

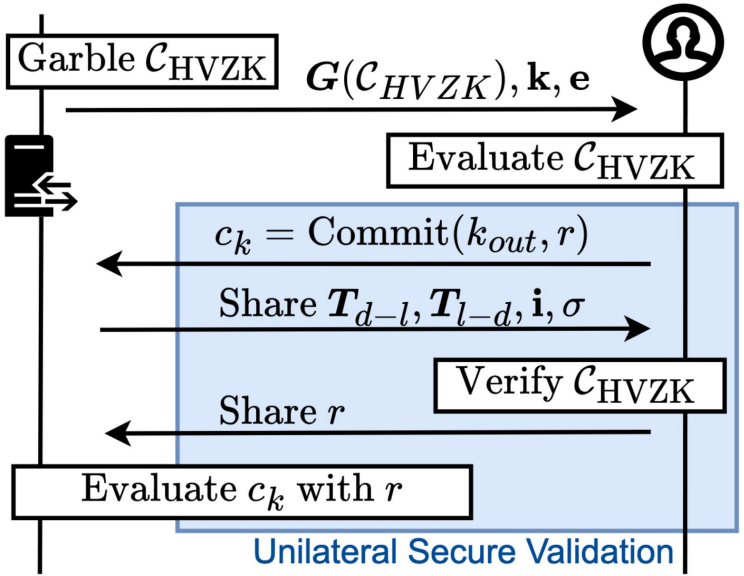
Protocol details: *Janus*

- **Insight 1:** The performance of 2PC AEAD tag computations deteriorates for larger record sizes
- **Insight 2:** Proof systems are not tailored to the arithmetic requirements and the privacy setting found in TLS oracles

Backup: Policy-driven Data Provenance

Protocol details: *Janus*

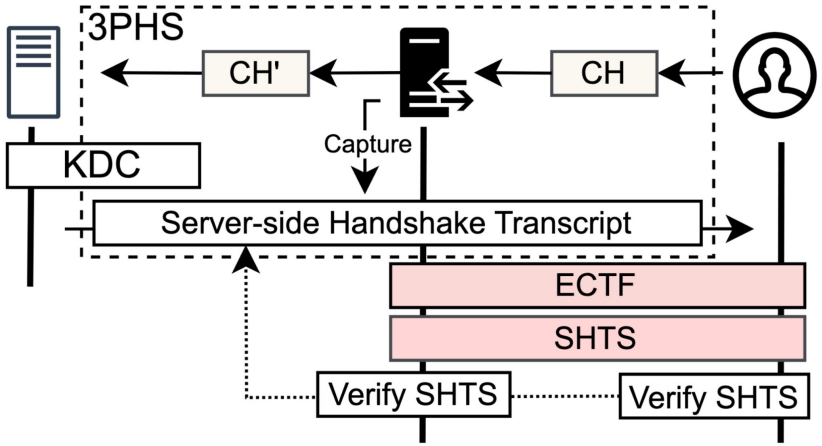
- Formalizing the Asymmetric Privacy Setting
- HVZK and Selective-failure Attacks
- Unilateral Secure Validation
- TLS Applicability



Backup: Policy-driven Data Provenance

Protocol details: *Janus*

- Pre-fetch for Immediate Server-side Handshake Transcript
- Compute and Disclose of SHTS
- Attacking SHTS Authenticity - SHTS Validation
- Optimized Garble-then-Prove (using SHTS authenticity)



Backup: Policy-driven Data Provenance

Protocol details: *Janus*

Circuit	Computation Trace
C_{XHTS}	DHE=s_1+s_2; DHE to XHTS
$C_{(\text{k,iv})}$	DHE=s_1+s_2; DHE to (kc_{XATS}, kv_{XATS}, iv_{XATS})
$C_{\text{CB}_{2+}}$	(kc_{XATS}, kv_{XATS}, iv_{XATS}) to CB_{2+}
C_t	(kc_{XATS}, kv_{XATS}, iv_{XATS}, ct) to t
C_{open}	DHE=s_1+s_2; DHE to SHTS; DHE to CB

$C_{\text{MtE}}(\mathbf{pt}, kc, kc^t\ ;\ kv, kv^t, \mathbf{ct}_\alpha{=}\mathbf{ct}[\text{end-3:}], \phi)$:

1. t' = HMAC(kc^t + kv^t, **pt**)
2. $\mathbf{ct}_\alpha$' = AES(kc + kv, t')
3. assert: $\mathbf{ct}_\alpha \stackrel{?}{=} \mathbf{ct}_\alpha$', $1 \stackrel{?}{=} f_\phi(\mathbf{pt})$

$C_{\text{AEAD}}(\mathbf{pt}, kc\ ;\ kv, \mathbf{ct}, \text{iv}, t_\alpha, \text{c}_k, \phi)$:

1. t_α' = [AES(kc + kv, **0**) , AES(kc + kv, iv||**1**)]
2. **ct**' = AES(kc + kv, **pt**); c_k' = commit(kc)
3. assert: $t_\alpha \stackrel{?}{=} t_\alpha$', $\text{c}_k \stackrel{?}{=} \text{c}_k$', $\mathbf{ct} \stackrel{?}{=} \mathbf{ct}$', $1 \stackrel{?}{=} f_\phi(\mathbf{pt})$

$C_{\mathbf{zkOpen}}(s_2, \mathbf{pt}\ ;\ s_1, \mathbf{I}^{\text{open}}, \mathbf{ct}, t, \text{SHTS}, \phi)$:

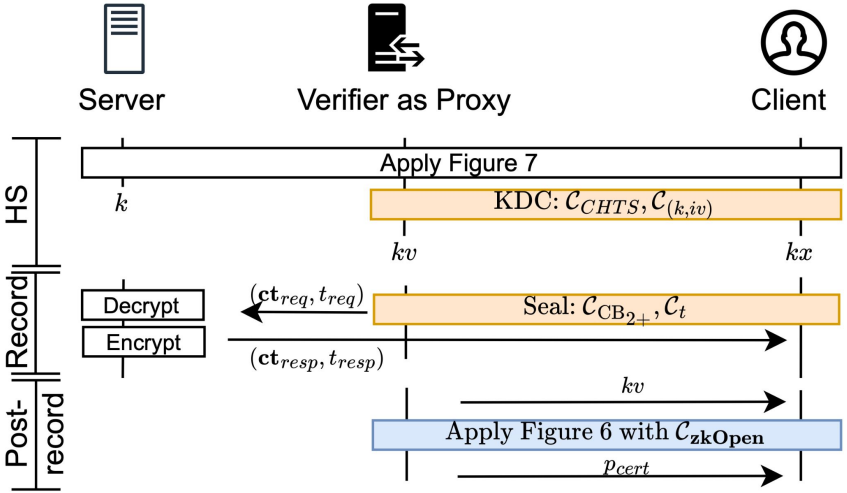
1. SHTS', **CB**' = $C_{\text{open}}(s_1, s_2, \mathbf{I}^{\text{open}})$
2. **ct**' = **CB**' ⊕ **pt**, t' = [CB_0' , CB_1']
3. assert: SHTS'$\stackrel{?}{=}$SHTS, **ct**'$\stackrel{?}{=}$ **ct**, $t' \stackrel{?}{=} t$, $1 \stackrel{?}{=} f_\phi(\mathbf{pt})$

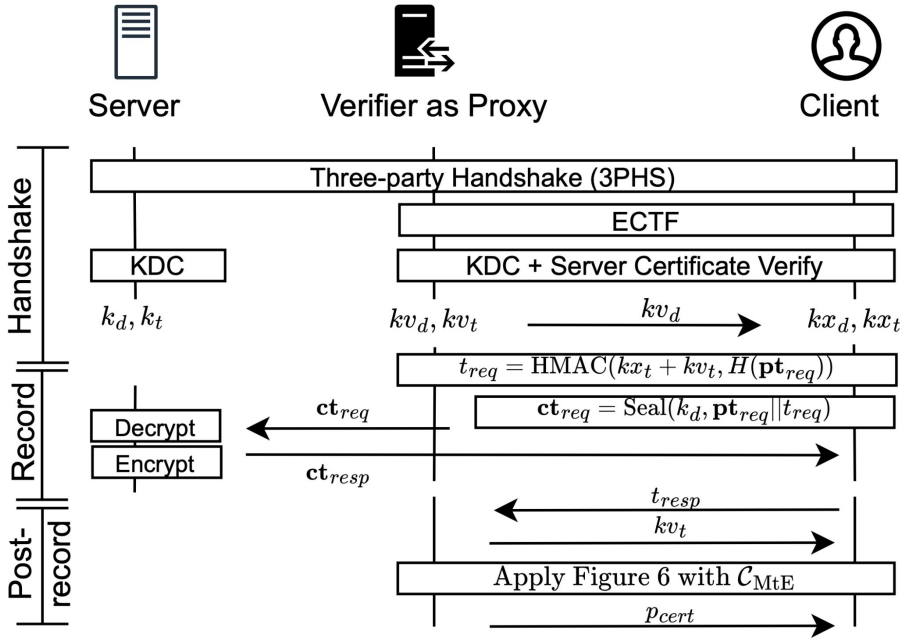
$C_{\mathbf{tpOpen}}(s_2\ ;\ s_1, \mathbf{I}^{\text{open}}, \mathbf{CB}, t, \text{SHTS})$:

1. SHTS', **CB**' = $C_{\text{open}}(s_1, s_2, \mathbf{I}^{\text{open}})$, t' = [CB_0', CB_1']
2. assert: SHTS'$\stackrel{?}{=}$SHTS, $t' \stackrel{?}{=} t$, **CB** $\stackrel{?}{=}$ **CB**'

Backup: Policy-driven Data Provenance

Protocol details: *Janus*

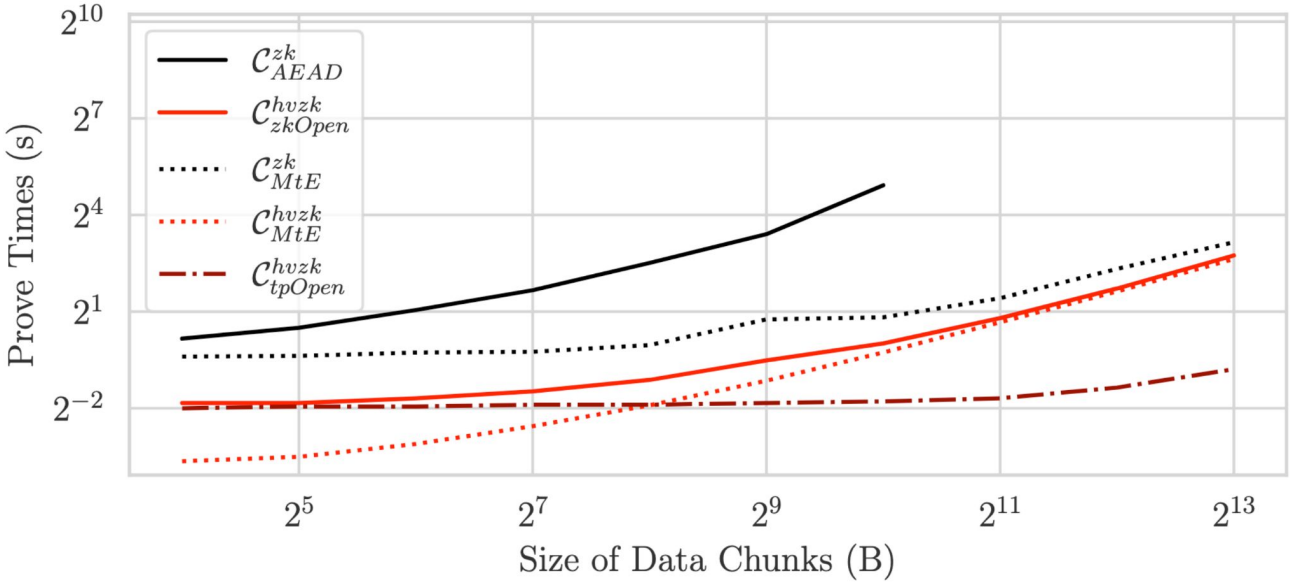




Backup: Policy-driven Data Provenance

Protocol details: *Janus*

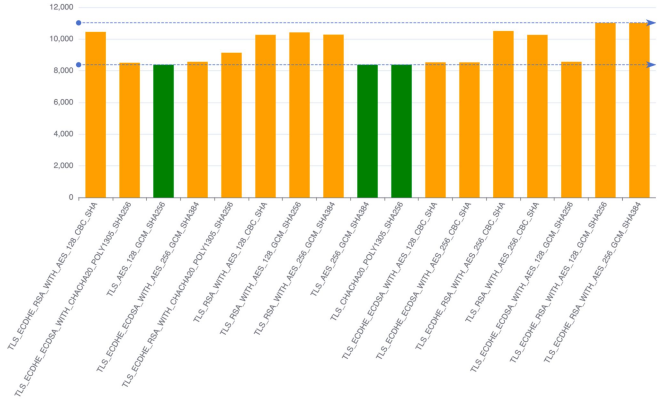
Mode	Variant	Protocols
12_GCM	AEAD	*TLSn*
12_CBC	MtE	*Deco*, *Janus12*
13_GCM	AEAD	*DiStefano*, *DecoProxy*, *Origo*, *Janus13*



Backup: Policy-driven Data Provenance

Protocol details: *Janus*

Paper	Type	Key Feature
TLSn/Deco	Notary	3PHS & 2PC-based TLS Client
Zombie	Proxy	Pad Commit
Origo	Proxy	2PC-free
DiStefano	Notary	Secure MtA & Browsing Privacy
XYWY23	Notary	Garble-then-prove & Pedersen Commit
Janus	Proxy	Secure 2PC-based hvZKP

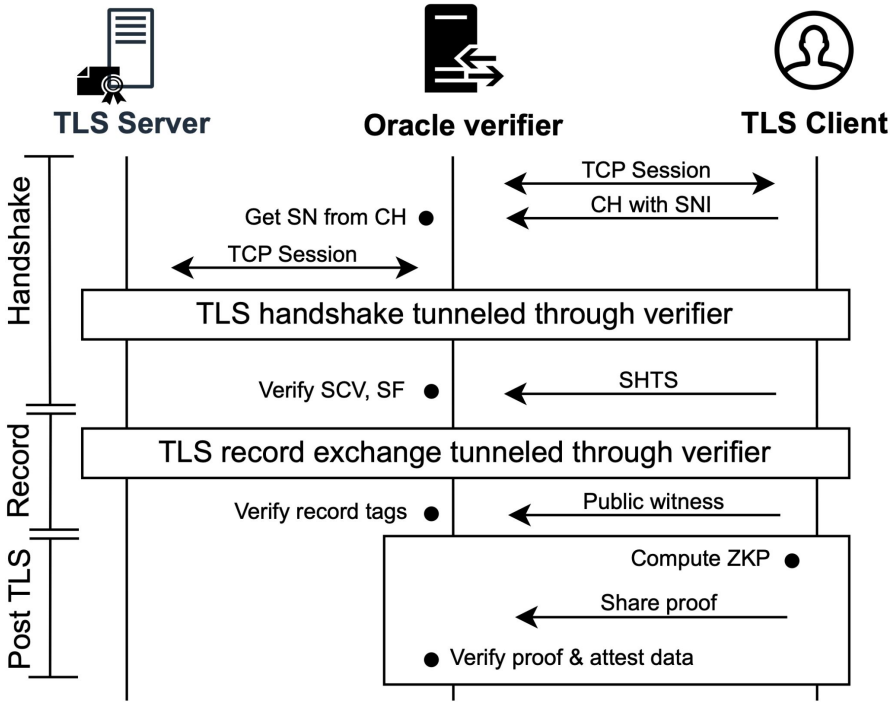


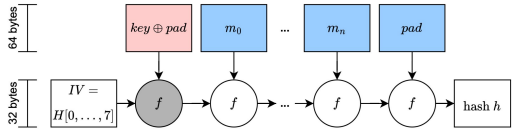
Protocol	Data	vTLS	Communication (kb)				Execution LAN (s) / WAN (s)			
			Offline	Handshake	Record	Post-record	Offline	Handshake	Record	Post-record
TLS	-	1.2	-	1.6	0.67	-	- / -	0.32 / 0.72	0.3 (ms) / 0.2	- / -
$TLSn_{tp}$	579b	1.2	178 (MB)	36 (MB)	40 (MB)	0.57	3.3 / 17.54	1.18 / 4.46	1.12 / 4.52	0.76 / 0.9
$Deco_{zk}$	32b	1.2	523.51 (MB)	294.52	150.55	0.23	14.38 / 56.26	0.94 / 2.36	0.68 / 1.59	0.76 / 0.86
$'Janus12_{zk}$	32b	1.2	415.22 (MB)	"	"	28.13	2.9 / 36.1	" / "	" / "	0.08 / 0.23
TLS	-	1.3	-	1.42	0.71	-	- / -	0.36 / 0.76	0.49 (ms) / 0.2	- / -
$*Origo_{zk}$	32b	1.3	367.61 (MB)	"	"	0.24	29.16 / 58.56	" / "	" / "	1.26 / 1.36
$*DecoProxy_{zk}$	32b	1.3	578.47 (MB)	307.7	"	0.27	24.34 / 70.61	0.95 / 2.37	" / "	1.35 / 1.45
DiStefano	256b	1.3	220.484 (MB)	343.42	48.82	-	5.85 / 23.48	0.43 / 0.85	0.12 / 0.32	- / -
$'Janus13_{tp}$	2.2kb	1.3	305.17 (MB)	113.8	984	583	1.99 / 26.4	0.51 / 0.91	1.04 / 1.51	0.46 / 0.6
$'Janus13_{zk}$	2.2kb	1.3	406.29 (MB)	"	"	2 (MB)	2.63 / 35.13	" / "	" / "	2.08 / 2.34

Backup: Policy-driven Data Provenance

Protocol details: *Origo*

- Proxy mode, no 2PC.
- Commitment attack (finds two messages $m_1 \neq m_2$ and two keys $k_1 \neq k_2$ such that encrypting m1 under k_1 and encrypting m_2 under k_2 yield the same ciphertext ct and tag t)
- ZKP key derivation





Backup: Policy-driven Data Provenance

Protocol details: *Origo*

Origo ZKP Circuit (dHS^{in}, pt^i ; $\mathbf{H_0}$, $\mathbf{H_2}$, $\mathbf{H_3}$, **SHTS**^{in}, $\mathbf{h}^{HS,opad}$, **MS**^{in}, **SATS**^{in}, **SATK**^{in}, $\mathbf{CB}_0^i$, $\mathbf{CB}_1^i$, **sIV**^{i}, **ct**^{i}, ϕ) :

1. tk_{sapp}, SHTS^{in} ← KDC(dHS^{in}, $\mathbf{H_0}$, $\mathbf{H_2}$, $\mathbf{H_3}$, $\mathbf{h}^{HS,opad}$, **MS**^{in}, **SATS**^{in}, **SATK**^{in})
2. ct^i ← AES128_GCM(**sIV**^{i}, pt^i, tk_{sapp})
3. $\mathbf{CB}_0^i$, $\mathbf{CB}_1^i$ ← AES128(**sIV**^{i}, tk_{sapp})
4. Assert if ct$^i \stackrel{?}{=}$ **ct**^{i} and if $\mathrm{CB}_0^i \stackrel{?}{=} \mathbf{CB}_0^i$ and if $\mathrm{CB}_1^i \stackrel{?}{=} \mathbf{CB}_1^i$ and if SHTS$^{in} \stackrel{?}{=}$ **SHTS**^{in}
5. Assert if $1 \stackrel{?}{=} f_\phi$(pt^i)

ZKP Circuit: Key Derivation(HS; $\mathbf{H_2}$, $\mathbf{H_3}$, **SHTS**)

1. **SHTS**← hkdf.exp(HS,"s hs traffic" || $\mathbf{H_2}$)
2. dHS ← hkdf.exp(HS,"derived", H(" "))
3. MS ← hkdf.ext(dHS, 0)
4. Client Application Traffic Secret (CATS) ← hkdf.exp(MS, "c ap traffic" || $\mathbf{H_3}$)
5. SATS ← hkdf.exp(MS, "s ap traffic" || $\mathbf{H_3}$)
6. tk_{capp} ← DeriveTK(CATS)
7. tk_{sapp} ← DeriveTK(SATS)

SHTS Verification(**SHTS**, $\mathbf{H_7}$, **SF**)

1. fk_S ← hkdf.exp(**SHTS**, "finished" || " ")
2. SF′ ← HMAC(fk_S, $\mathbf{H_7}$)
3. SF′ $\stackrel{?}{=}$ **SF**
4. ok $\stackrel{?}{=}$ verifyCertificate()

Initialize:
l_0 = "tls13 derived"; l_2 = "tls13 s hs traffic"; l_3 = "tls13 s ap traffic"
l_f = "tls13 finished"; l_k = "tls13 key"; l_{iv} = "tls13 iv";
$m^{\mathbf{H_2}} = 32 \,||\, len(l_2) \,||\, l_2 \,||\, len(\mathbf{H_2}) \,||\, \mathbf{H_2} \,||\, 1$; $m^{\mathbf{H_3}} = 32 \,||\, len(l_3) \,||\, l_3 \,||\, len(\mathbf{H_3}) \,||\, \mathbf{H_3} \,||\, 1$;
$m^{\mathbf{H_0}} = 32 \,||\, len(l_0) \,||\, l_0 \,||\, len(\mathbf{H_0}) \,||\, \mathbf{H_0} \,||\, 1$; $m_{iv} = 12 \,||\, len(l_{iv}) \,||\, l_{iv} \,||\, 0 \,||\, 1$;
$m_f = 32 \,||\, len(l_f) \,||\, l_f \,||\, 0 \,||\, 1$; $m_k = 16 \,||\, len(l_k) \,||\, l_k \,||\, 0 \,||\, 1$;

Key Derivation Trace:

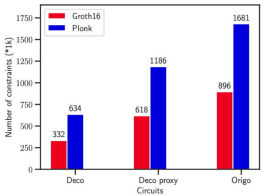
1. p: $h^{HS,ipad}, l_x = f$ (IV, 0, HS ⊕ $ipad$); **SHTS**^{in}, _ = f ($h^{HS,ipad}$, l_x, $m^{\mathbf{H_2}}$)
2. p: $\mathbf{h}^{HS,opad}, l = f$ (IV, 0, HS ⊕ $opad$)
3. v: SHTS , _ = f ($\mathbf{h}^{HS,opad}$, l, **SHTS**^{in}); $fk_S{}^{in}$, _ = f (f (IV, 0, SHTS ⊕ $ipad$), m_f)
4. v: fk_S , _ = f (f (IV, 0, SHTS ⊕ $opad$), $fk_S{}^{in}$); SF′^{in}, _ = f (f (IV, 0, fk_S ⊕ $ipad$), $\mathbf{H_7}$ || 1)
5. v: SF′ , _ = f (f (IV, 0, fk_S ⊕ $opad$), SF′^{in}); SF′ $\stackrel{?}{=}$ **SF**

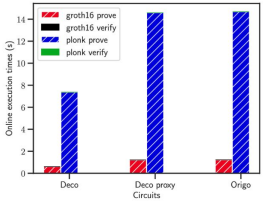
10. p: dHS$^{in}, l = f$ ($h^{HS,ipad}$, l_x, $m^{\mathbf{H_0}}$)
11. zk: dHS , _ = f ($\mathbf{h}^{HS,opad}$, l, dHS^{in})
12. p: $\mathbf{h}^{dHS,ipad}, l = f$ (IV, 0, dHS ⊕ $ipad$)
13. v: **MS**^{in}, _ = f ($\mathbf{h}^{dHS,ipad}$, l, 0bytes)
14. zk: MS , _ = f (f (IV, 0, dHS ⊕ $opad$), **MS**^{in})
15. p: $\mathbf{h}^{MS,ipad}, l = f$ (IV, 0, MS ⊕ $ipad$)
16. v: **SATS**^{in}, _ = f ($\mathbf{h}^{MS,ipad}$, l, $m^{\mathbf{H_3}}$)
17. zk: SATS , _ = f (f (IV, 0, MS ⊕ $opad$), **SATS**^{in})

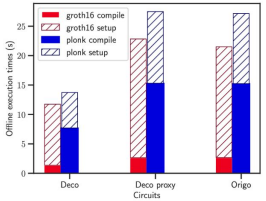
18. p: $\mathbf{h}^{SATS,ipad}, l = f$ (IV, 0, SATS ⊕ $ipad$)
19. v: **SATK**^{in}, _ = f ($\mathbf{h}^{SATS,ipad}$, l, m_k)
20. zk: $h^{SATS,opad}, l = f$ (IV, 0, SATS ⊕ $opad$)
21. zk: tk_{sapp} , _ = f ($h^{SATS,opad}$, l, **SATK**^{in})[:16]
22. v: **sIV**^{in}, _ = f ($\mathbf{h}^{SATS,ipad}$, l, m_{iv})
23. p: **sIV**, _ = f ($h^{SATS,opad}$, l, **sIV**^{in})[:12]

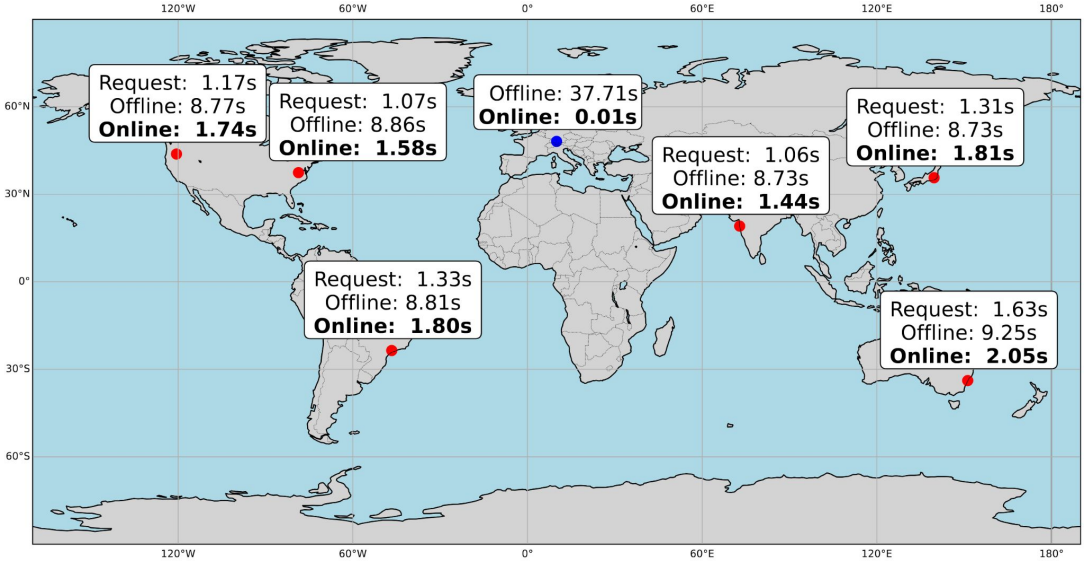
Backup: Policy-driven Data Provenance

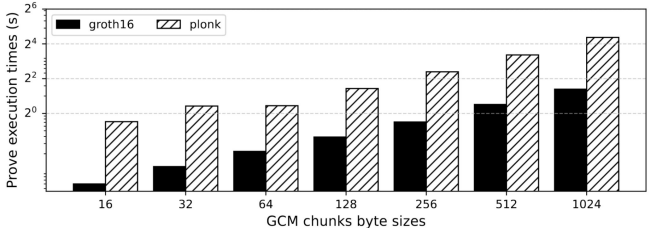
Protocol details: *Origo*

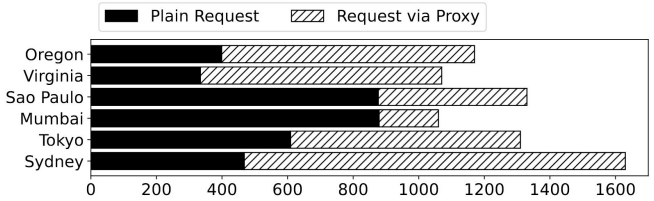






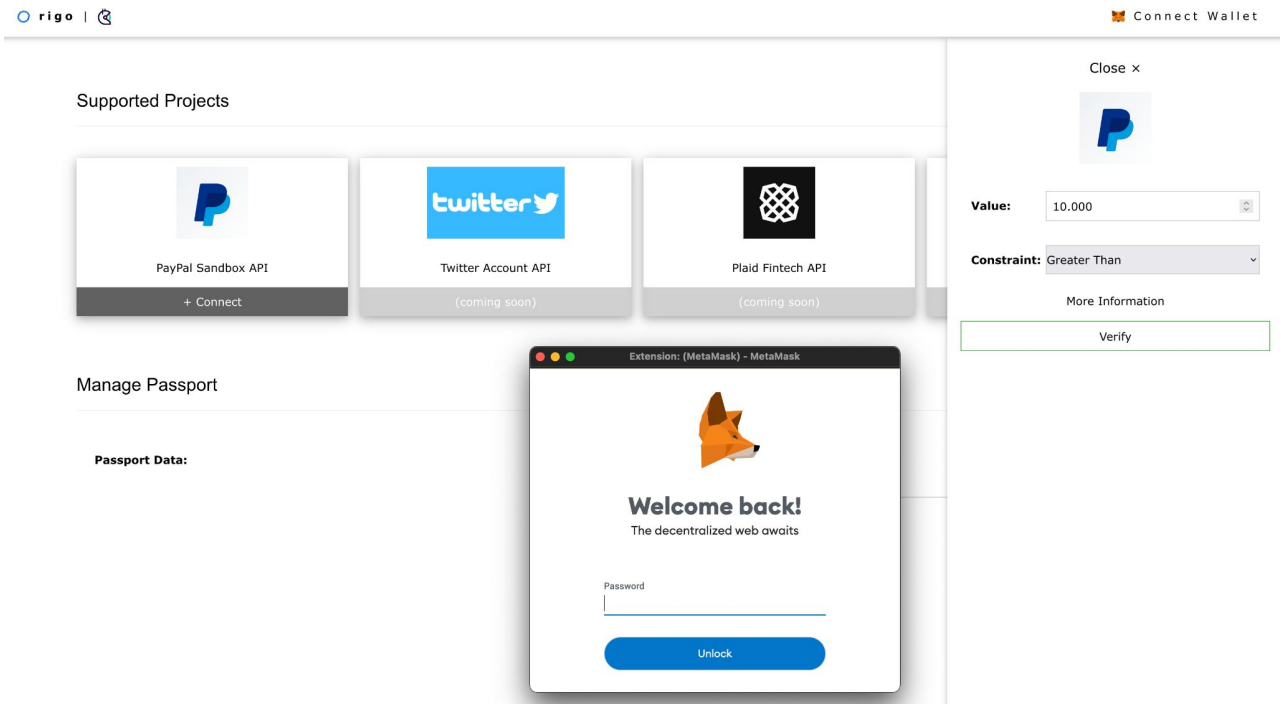






Backup: Policy-driven Data Provenance

Protocol details: *zkGen*



Backup: Policy-driven Data Provenance

Protocol details: *zkGen*

“paypal bank balance > 100$”

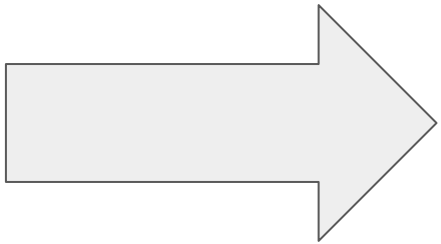
```
commit(r, secret) == c
k = HKDF(secret)
plaintext = decrypt(k, ciphertext)
substring = find(plaintext, pattern)
bank_balance = convert(substring)
bank_balance > 100
```

A simple **statement**

A complex **circuit** (#constraints=1.6million)

Protocol details: *zkGen*

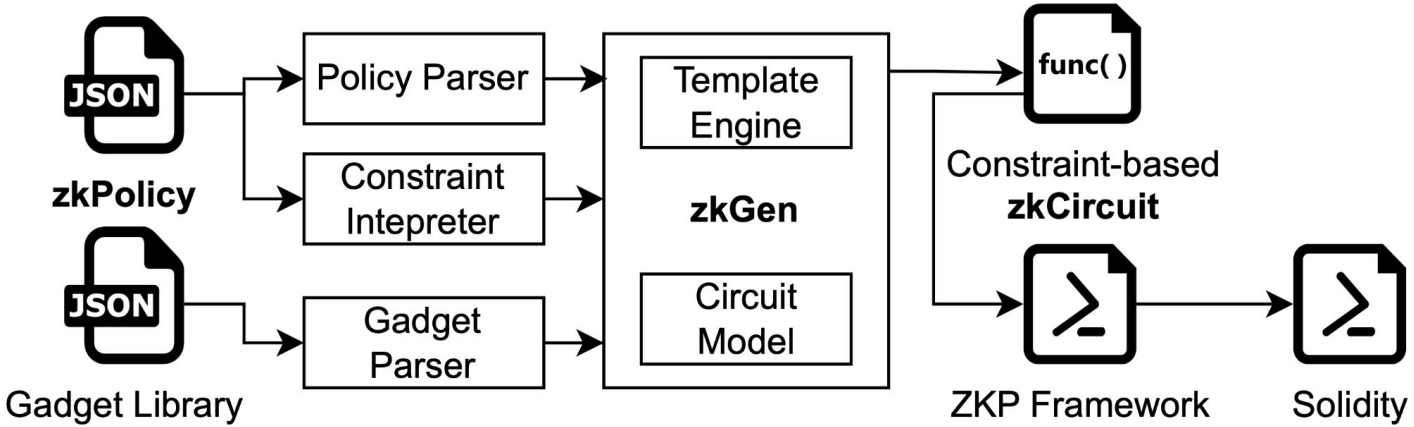
- zkPolicy Language
- zkGen Transpiler



Concise description of compliant and private computations with a **fraction** of code lines.

zkPolicy	LOC policy	LOC circuit	Transpile time
TLS commit	22	975	2.38 ms
Age commit	20	85	1.70 ms

Protocol details: *zkGen*



Backup: Policy-driven Data Provenance

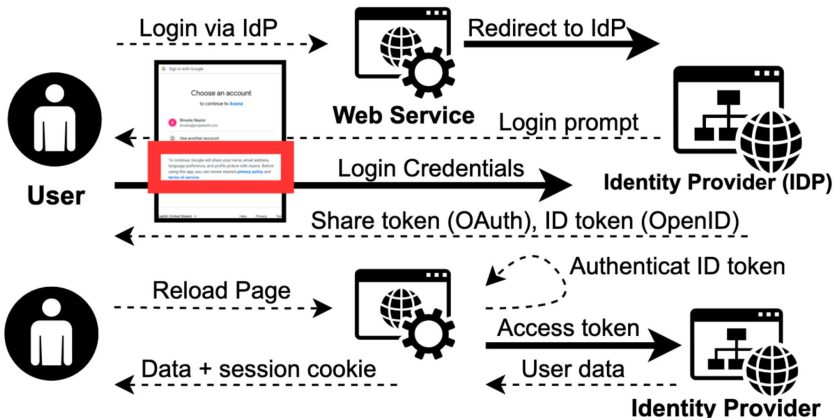
Protocol details: *zkGen*

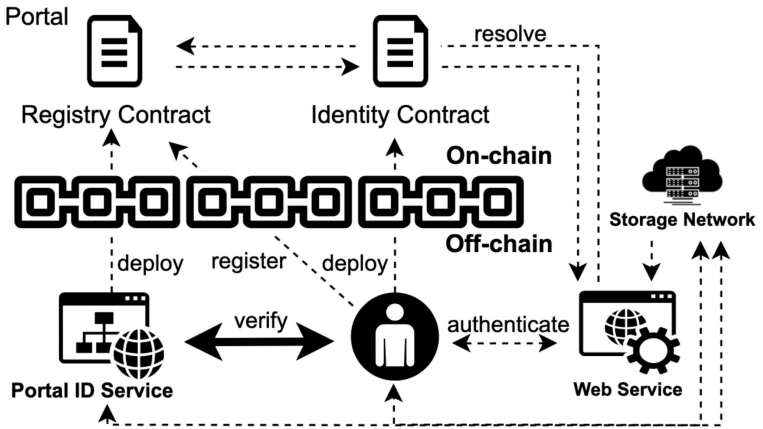
```
{"name": "zkPolicy_tls13openSessionCommit",
 "relations":[{
   "value": {
     "type": "s-string",
     "size": 5,
     "protection": {
       "algorithm": "secure_channels:openRecord_TLS13AES128GCMSHA256",
       "parameter": "value"
     }
   },
   "number": {"type": "p-integer"}
 }],
 "constraints": [
   "0:value->-0:number",
   "0:value:protection:algorithm:key-=-commitments:mimc:message"
 ]
}
```

```
{"commitments": {
 "mimc": {
   "commit_string":
     {"type": "p-string", "size": 32},
   "witness":
     {"type": "s-string", "size": -1},
   "message":
     {"type": "s-string", "size": -1}
}}}
```

Backup: Sovereign Identity & Data Privacy

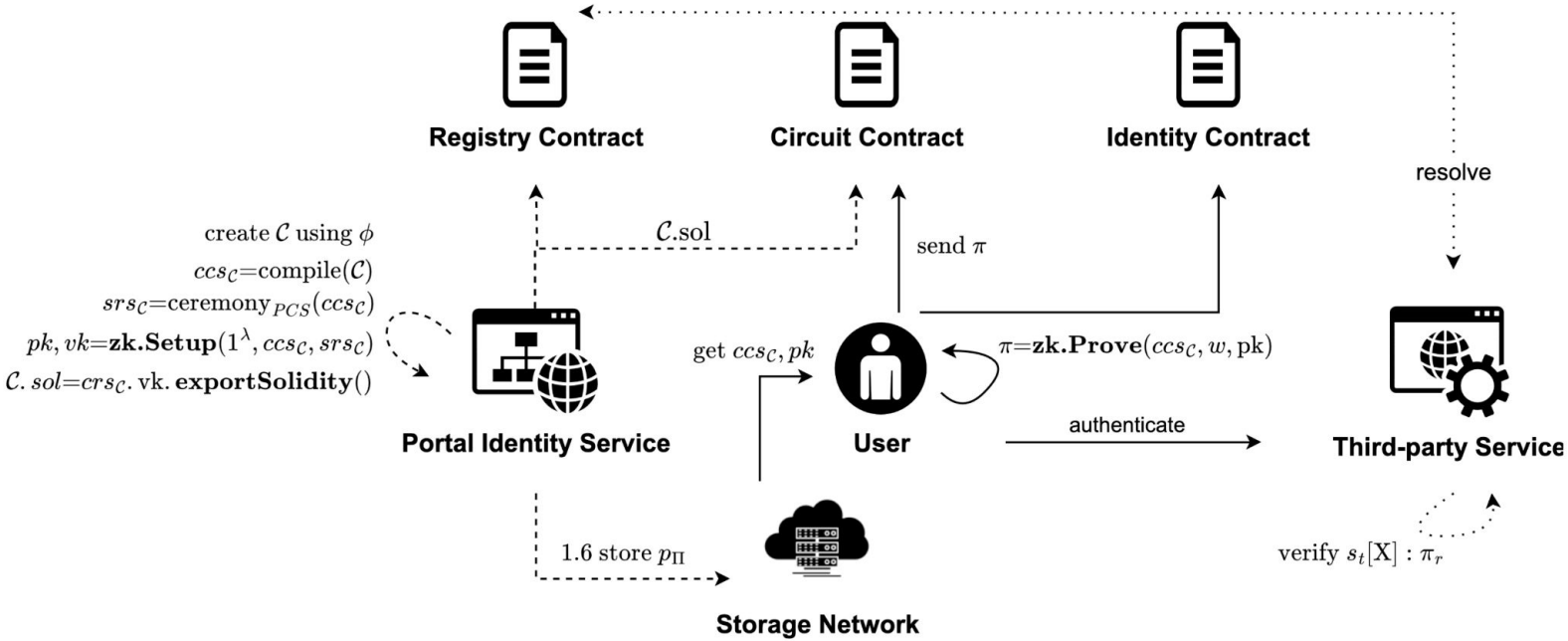
Protocol details: *Portal*





Backup: Sovereign Identity & Data Privacy

Protocol details: *Portal*



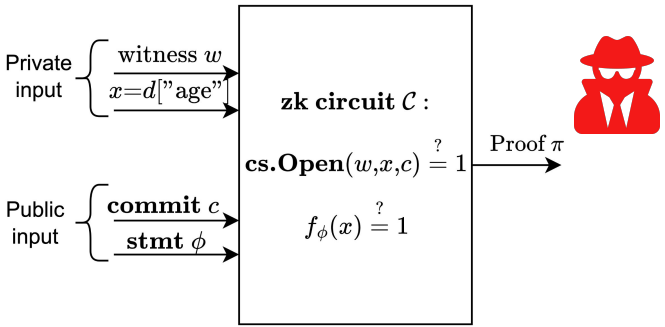
Backup: Sovereign Identity & Data Privacy

Protocol details: *Portal*

```
https://api.etherscan.io/api
   ?module=logs
   &action=getLogs
   &fromBlock=15073139
   &toBlock=15074139
   &address=0x59728544b08ab483533076417fbbb2fd0b17ce3a
   &topic0=0x27c4f0403323142b599832f26acd21c74a9e5b809f2215726e244a4ac588cd7d
   &topic0_1_opr=and
   &topic1=0x00000000000000000000000023581767a106ae21c074b2276d25e5c3e136a68b
   &page=1
   &offset=1000
   &apikey=YourApiKeyToken
```


```
{
   "status":"1",
   "message":"OK",
   "result":[
      {
         "address":"0x59728544b08ab483533076417fbbb2fd0b17ce3a",
         "topics":[
            "0x27c4f0403323142b599832f26acd21c74a9e5b809f2215726e244a4ac588cd7d",
            "0x00000000000000000000000023581767a106ae21c074b2276d25e5c3e136a68b",
            "0x000000000000000000000000000000000000000000000000000000000000236d",
            "0x00000000000000000000000c8a5592031f93debea5d9e67a396944ee01bb2ca"
         ],
         "data":"0x00000000000000000000000c02aaa39b223fe8d0a0e5c4f27ead9083c756cc200
         "blockNumber":"0xe60262",
         "timeStamp":"0x62c26caf",
         "gasPrice":"0x5e2d742c9",
         "gasUsed":"0xfb7f8",
         "logIndex":"0x4b",
         "transactionHash":"0x26fe1a0a403fd44ef11ee72f3b4ceff590b6ea533684cb279cb4242
         "transactionIndex":"0x39"
      },
```

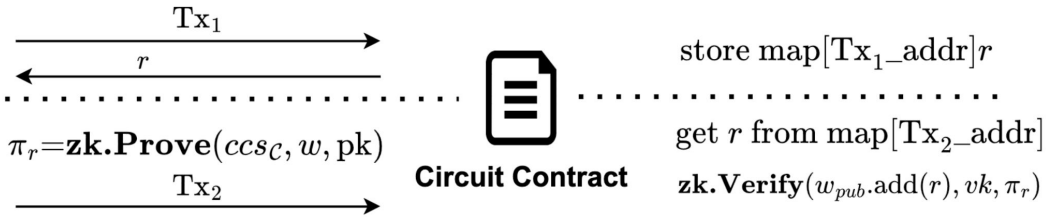

- Public claims
  - Users remain accountable if any misbehaviour is detected
- Private claims
  - **Open issue**, which we solve

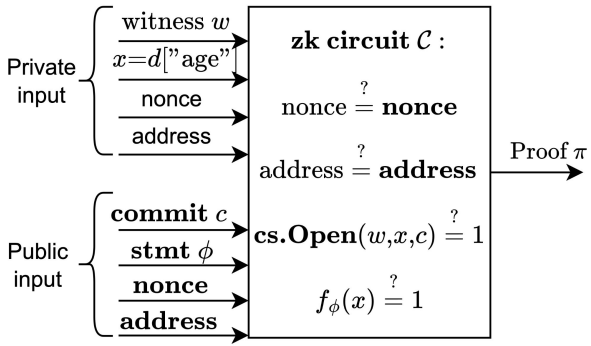


Protocol details: *Portal*

New transaction sequence

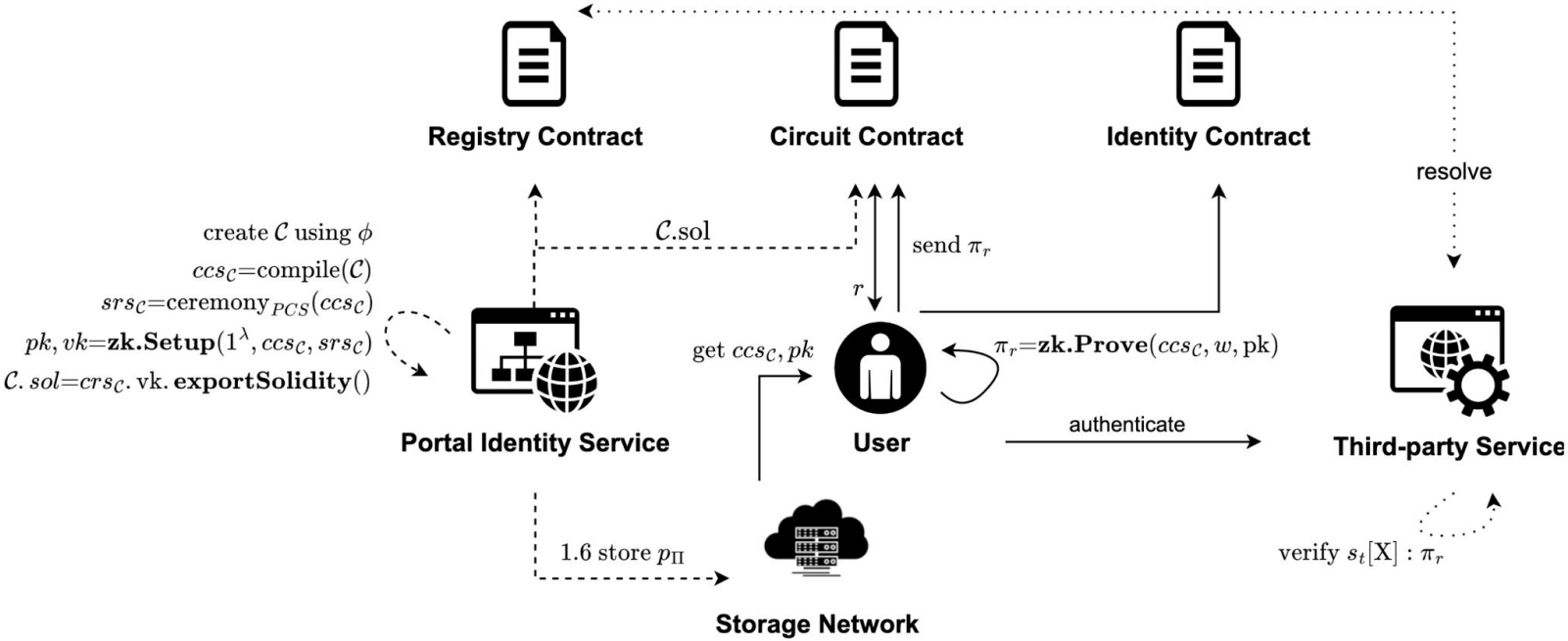
- Tx_1 samples randomness at contract and maps it to Tx_1 sender address
- Proof computation bound by Tx_1 transaction time
- Tx_2 sends proof $\boldsymbol{\pi}$ with additional circuit logic (asserts nonce & address)





Backup: Sovereign Identity & Data Privacy

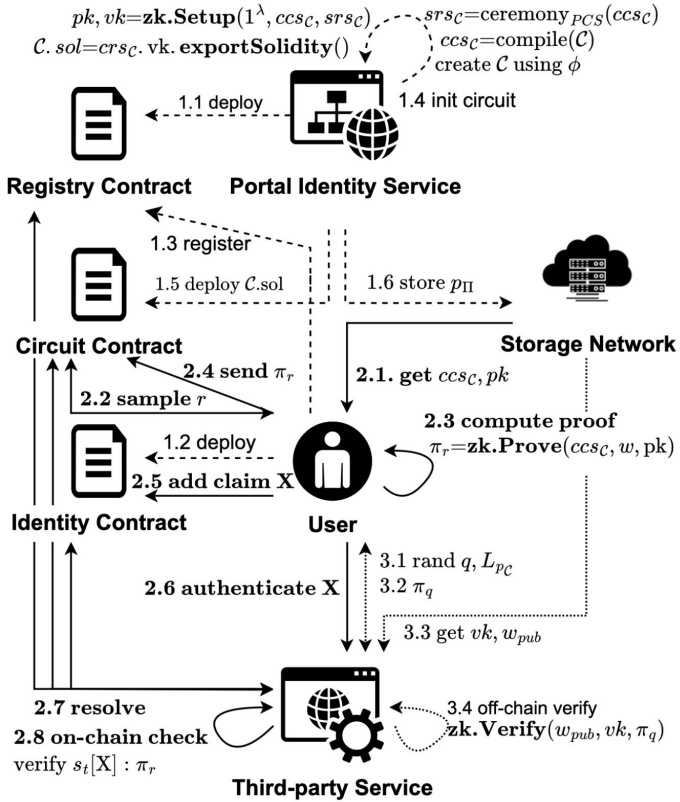
Protocol details: *Portal*



Backup: Sovereign Identity & Data Privacy

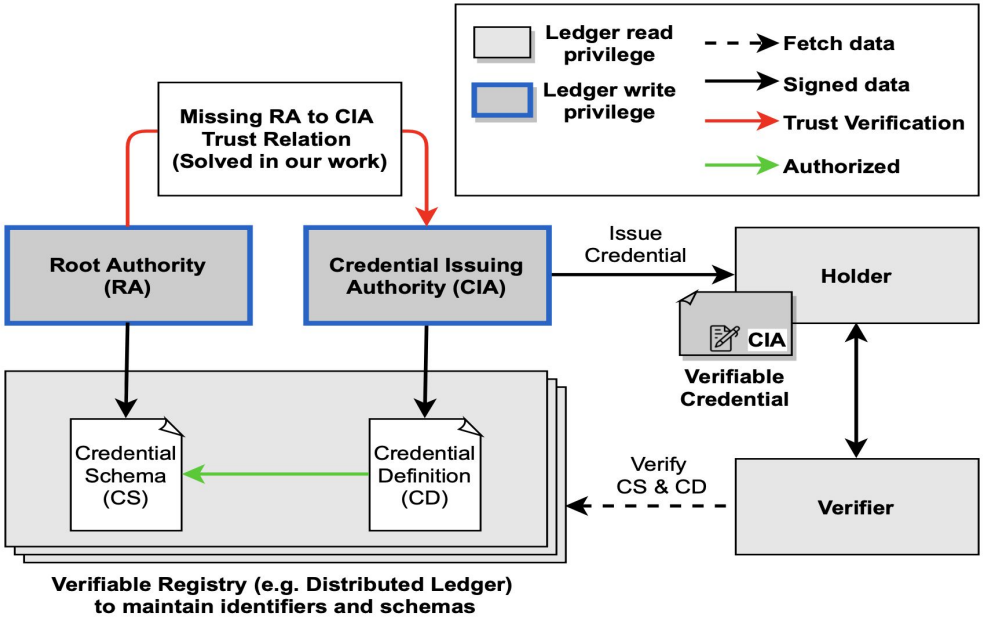
Protocol details: *Portal*

Tx / Call	Type	Cost (eth/$)	Time (ms)	Size (kB)
C^{reg}	deploy	4.1e-3/8.6	18	bc:6.5,tx:6.6
C^{id}	deploy	6.5e-3/13.5	10	bc:10,tx:10
$C^{\mathcal{C}_1}$	deploy	4.9e-3/10.2	**385**	bc:7.4,tx:12
set_$\mathcal{C}_1$	C^{reg}	8.4e-5/0.18	11	tx: 0.46
register	C^{reg}	7.4e-5/0.16	51	tx: 0.3
claim^{pub}	C^{id}	6.4e-05/0.13	3	tx: 0.48
sample	$C^{\mathcal{C}_1}$	6.6e-05/0.14	6	tx: 0.1
verify_π	$C^{\mathcal{C}_1}$	8.4e-4/**1.76**	**252**	tx: 1.20
claim^{priv}	C^{id}	3.9e-4/0.82	21	tx: 0.68
setup$^{\mathcal{C}_1}_{\text{plonk}}$	off-chain	-	1029	p_Π: 7430
prove$^{\mathcal{C}_1}_{\text{plonk}}$	off-chain	-	195	π: 0.552
set/get$^{p_\Pi}_{\text{IPFS}}$	off-chain	-	631 / 66	7430
get$^{W/n/\mathcal{C}_1}$	off-chain	-	10/6.2/4.8	42/78/130



Backup: Sovereign Identity & Data Privacy

Protocol details: *A-PoA*



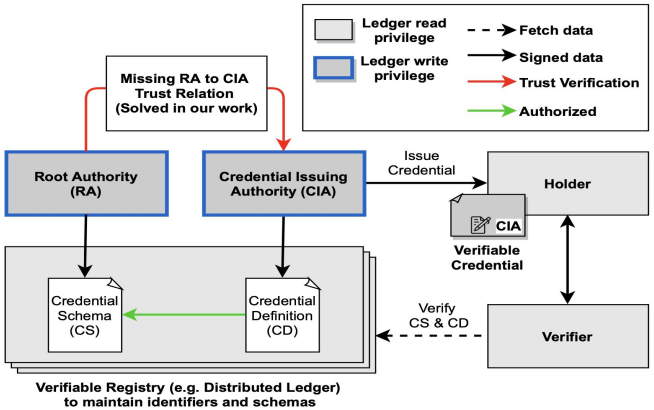
Backup: Sovereign Identity & Data Privacy

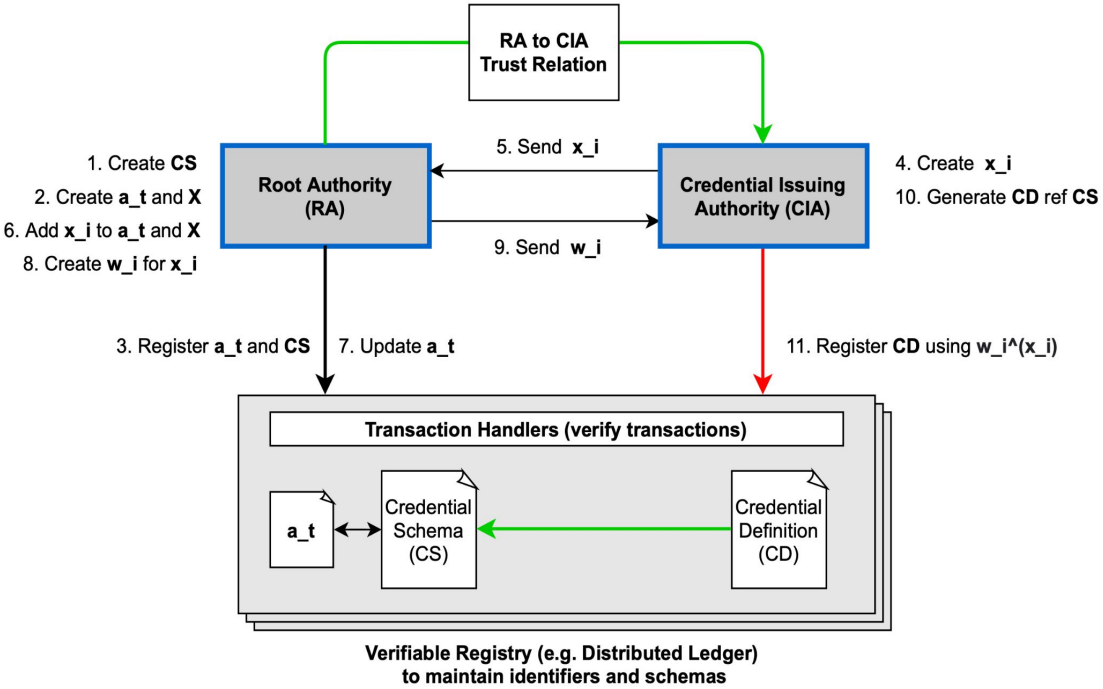
Protocol details: *A-PoA*

- Initialize accumulator: $a_t = (g_{\text{acc}})^2 \mod n$
- Add element to the accumulator: $a_{t+1} = a_t^{x_i} \mod n$
- Calculate element witness pair (x_i, w_i): $w_i = a_t^{X_t \setminus \{x_i\}} \mod n$
- Verify accumulator membership of x_i: $w_t^{x_i} \mod n == a_{t+1}$
- Revoke accumulator element x_i: $a_{t+1} = a_t^{x_i^{-1} \mod \phi(n)} \mod n$, with $\phi(n) = (p-1) \cdot (q-1)$
- Requirement to update all witness values after addition or revocation.

Backup: Sovereign Identity & Data Privacy

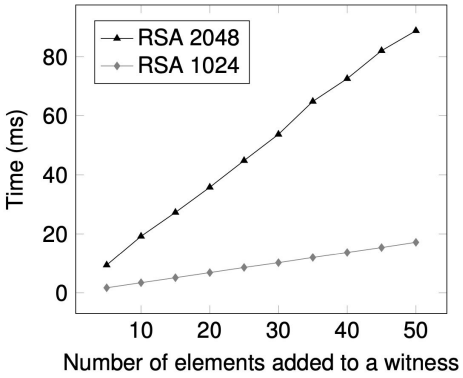
Protocol details: *A-PoA*

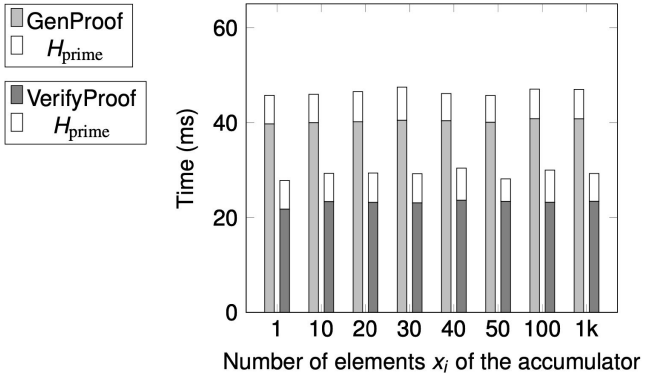




Protocol details: *A-PoA*

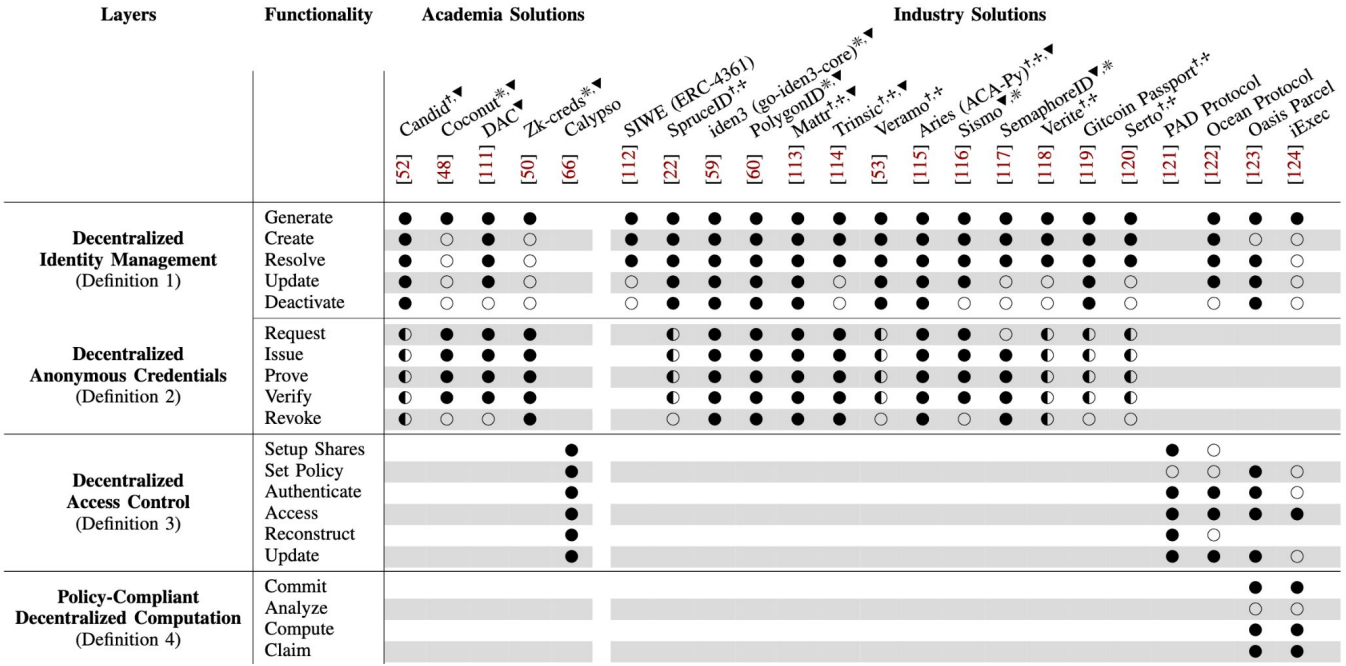
Protocol	Function	Time (ms)	COM	Big $\mathcal{O}$
Authorization	Prime Gen.	309.81	k	$\mathcal{O}(n)$
	Acc. Gen.	88.80	1	$\mathcal{O}(1)$
	Wit. Gen.	4383.60	k	$\mathcal{O}(n)$
Authentication	GenProof	40.06	1	$\mathcal{O}(1)$
	VerifyProof	23.42	0	
Revocation	Acc. Revoke	2244.85	(0-k)	$\mathcal{O}(n)$
$\sum_{\text{Authorization}}$	N/A	4782.21	$2k+1$	$\mathcal{O}(n)$
$\sum_{\text{Authentication}}$	N/A	63.48	1	$\mathcal{O}(1)$
$\sum_{\text{Revocation}}$	N/A	2244.85	(0-(k-1))	$\mathcal{O}(n)$





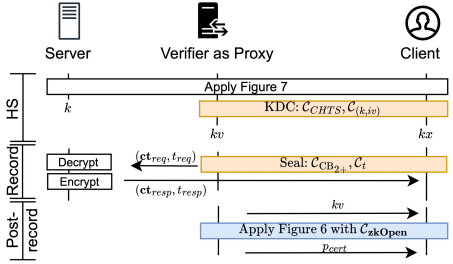
Backup: Sovereign Identity & Data Privacy

Protocol details: *SoK*



Layers	Functionality	Academia Solutions					Industry Solutions																
		Candid†,◀ [52]	Coconut*,◀ [48]	DAC◀ [111]	Zk-creds*,◀ [50]	Calypso [66]	SIWE (ERC-4361) [112]	SpruceID†,* [22]	iden3 (go-iden3-core)*,◀ [59]	PolygonID*,◀ [60]	Mattr†,*,◀ [113]	Trinsic†,*,◀ [114]	Verano†,* [53]	Aries (ACA-Py)†,*,◀ [115]	Sismo◀,* [116]	SemaphoreID◀,* [117]	Verite†,* [118]	Gitcoin Passport†,* [119]	Serto†,* [120]	PAD Protocol [121]	Ocean Protocol [122]	Oasis Parcel [123]	iExec [124]
Decentralized Identity Management (Definition 1)	Generate	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●
	Create	●	○	●	○		●	●	●	●	●	●	●	●	●	●	●	●	●		●	○	○
	Resolve	●	○	●	○		●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	○
	Update	●	○	●	○		○	●	●	●	●	○	●	●	●	○	○	●	○		●	●	○
	Deactivate	●	○	○	○		○	●	●	●	●	○	●	●	○	○	○	●	○		○	●	○
Decentralized Anonymous Credentials (Definition 2)	Request	◐	●	●	●			◐	●	●	●	●	◐	●	●	○	◐	◐	◐				
	Issue	◐	●	●	●			◐	●	●	●	●	◐	●	●	●	◐	◐	◐				
	Prove	◐	●	●	●			◐	●	●	●	●	◐	●	●	●	◐	◐	◐				
	Verify	◐	●	●	●			◐	●	●	●	●	◐	●	●	●	◐	◐	◐				
	Revoke	◐	○	○	●			○	●	●	●	●	○	●	○	●	◐	○	○				
Decentralized Access Control (Definition 3)	Setup Shares					●														●	○		
	Set Policy					●														○	○	●	○
	Authenticate					●														●	●	●	○
	Access					●														●	●	●	●
	Reconstruct					●														●	○		
	Update					●														●	●	●	○
Policy-Compliant Decentralized Computation (Definition 4)	Commit																					●	●
	Analyze																					○	○
	Compute																					●	●
	Claim																					●	●

Backup
Additional Figures



zkPolicy	LOC policy	LOC circuit	Transpile Time
TLS 1.3 commit	22	975	2.38 ms
Age commit	20	85	1.70 ms