Lecture 7: “zkTLS” or Provence of TLS Data

CPSC 444/544, 2026 Spring

Fan Zhang

Yale University

New Assignments

- See Canvas for HW2

- Assignments related to Final Project are posted too.
- They only apply to students enrolled in 5440.
- We will talk about projects next week.

Example #1: Age Verification

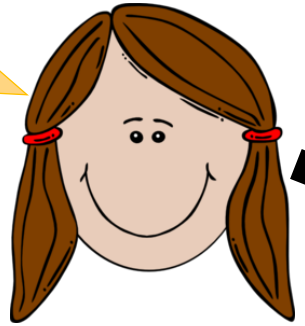
Age > 21 only

Verifier

But I can't see your data!

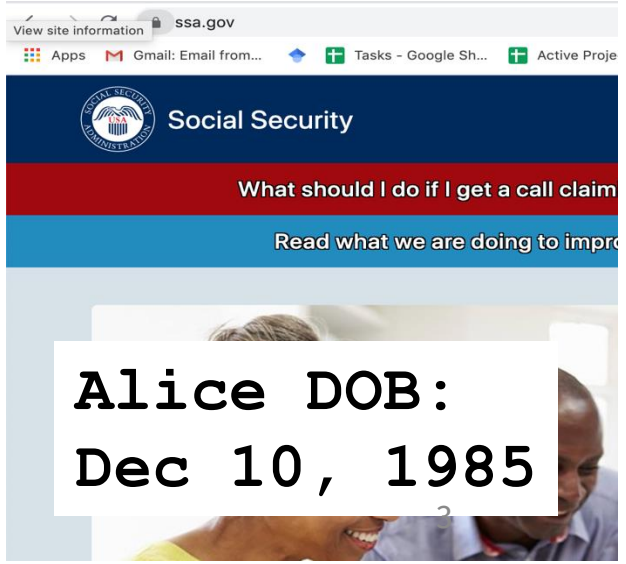
I am over 21 according to ssa.gov

Prover



Talk over the Internet

Login: password



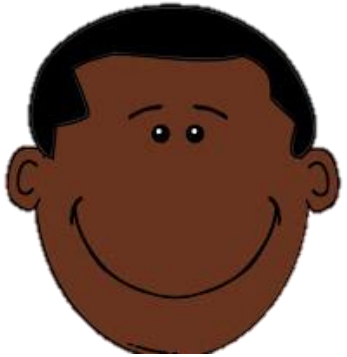
Alice's two goals

- Convince the verifier
- Doesn't leak her exact DoB

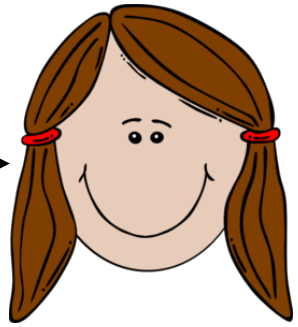
Goal: Proving *statements* about TLS-protected data

My DoB (Y/M/D) on SSA.gov satisfies that 2022 – Y > 18.

Verifier



Prover



https://

Two goals

- **Integrity**: Prover can't fool the verifier.
- **Privacy**: Verifier doesn't learn more than the statement being true.

Goal: Proving *statements* about TLS-protected data

My DoB (Y/M/D) on SSA.gov satisfies that 2022 – Y > 18.

Verifier

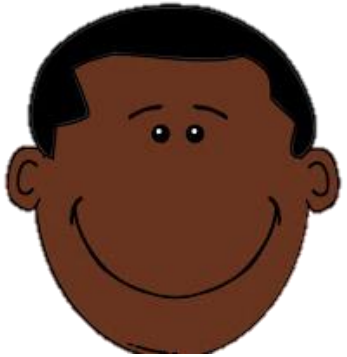
Prover

Let's rule out some bad ideas:
#1: Prover to send over the SSA.gov password
#2: Prover to send over a screenshot of SSA.gov

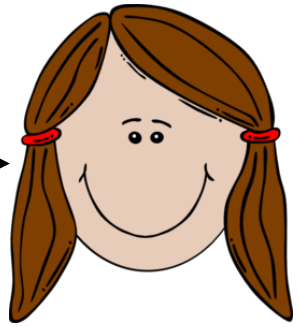
Goal: Proving *statements* about TLS-protected data

My DoB (Y/M/D) on SSA.gov satisfies that 2022 – Y > 18.

Verifier

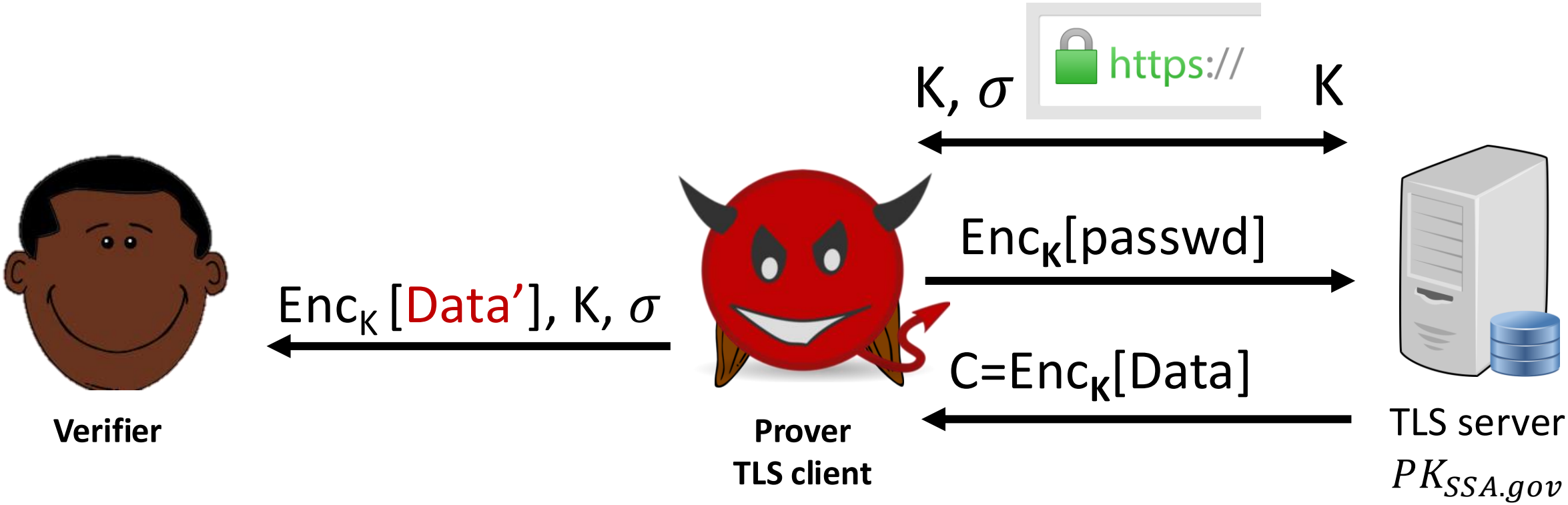


Prover



Question: What about sending over the TLS encrypted data?

Problem: TLS doesn't support provenance



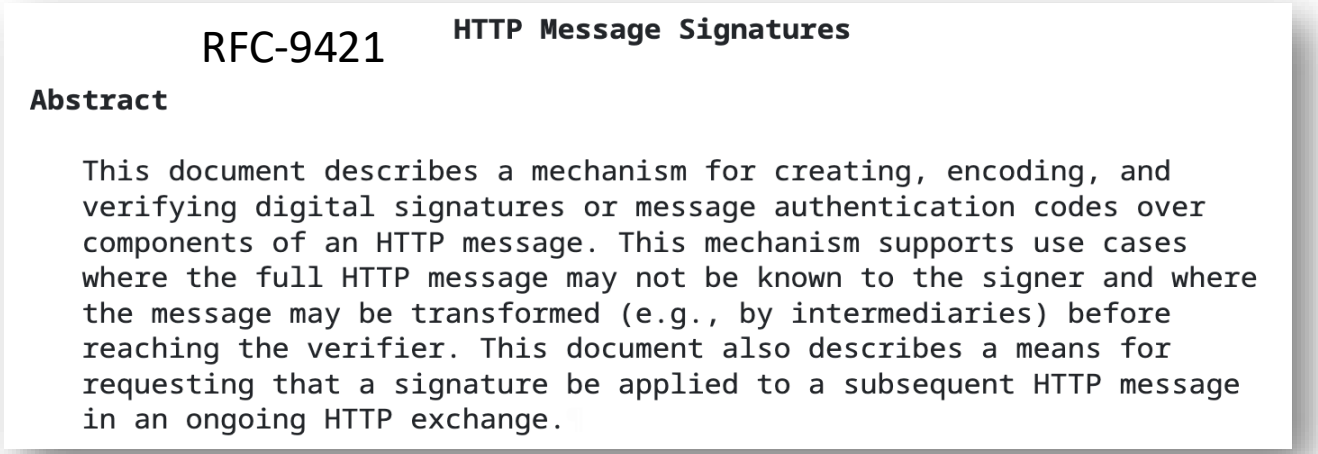
TLS doesn't sign the data!

No better than sending a screenshot.

How about we just change the web?

- E.g., Change TLS to sign the data (TLS-N) or add signature to HTTP messages (e.g., RFC-9421)
- Challenges: adoption barrier
  - making deniability impossible

Ritzdorf, Hubert, et al. "TLS-N: Non-repudiation over TLS Enabling Ubiquitous Content Signing." In *NDSS,* 2018.

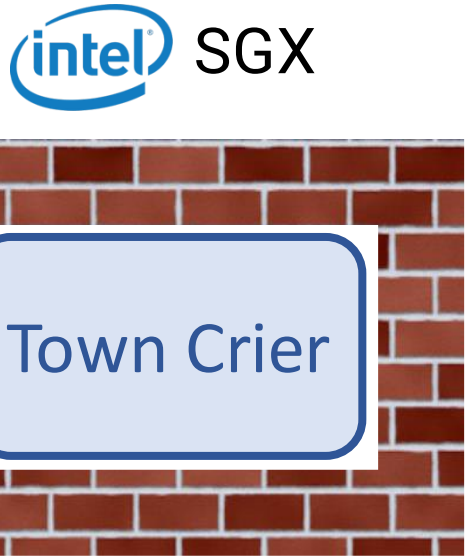

RFC-9421 HTTP Message Signatures

Abstract

This document describes a mechanism for creating, encoding, and verifying digital signatures or message authentication codes over components of an HTTP message. This mechanism supports use cases where the full HTTP message may not be known to the signer and where the message may be transformed (e.g., by intermediaries) before reaching the verifier. This document also describes a means for requesting that a signature be applied to a subsequent HTTP message in an ongoing HTTP exchange.

We'll see two solutions in this class

- Unique feature: requires no changes to websites



Using **TEEs**
(to be covered later)

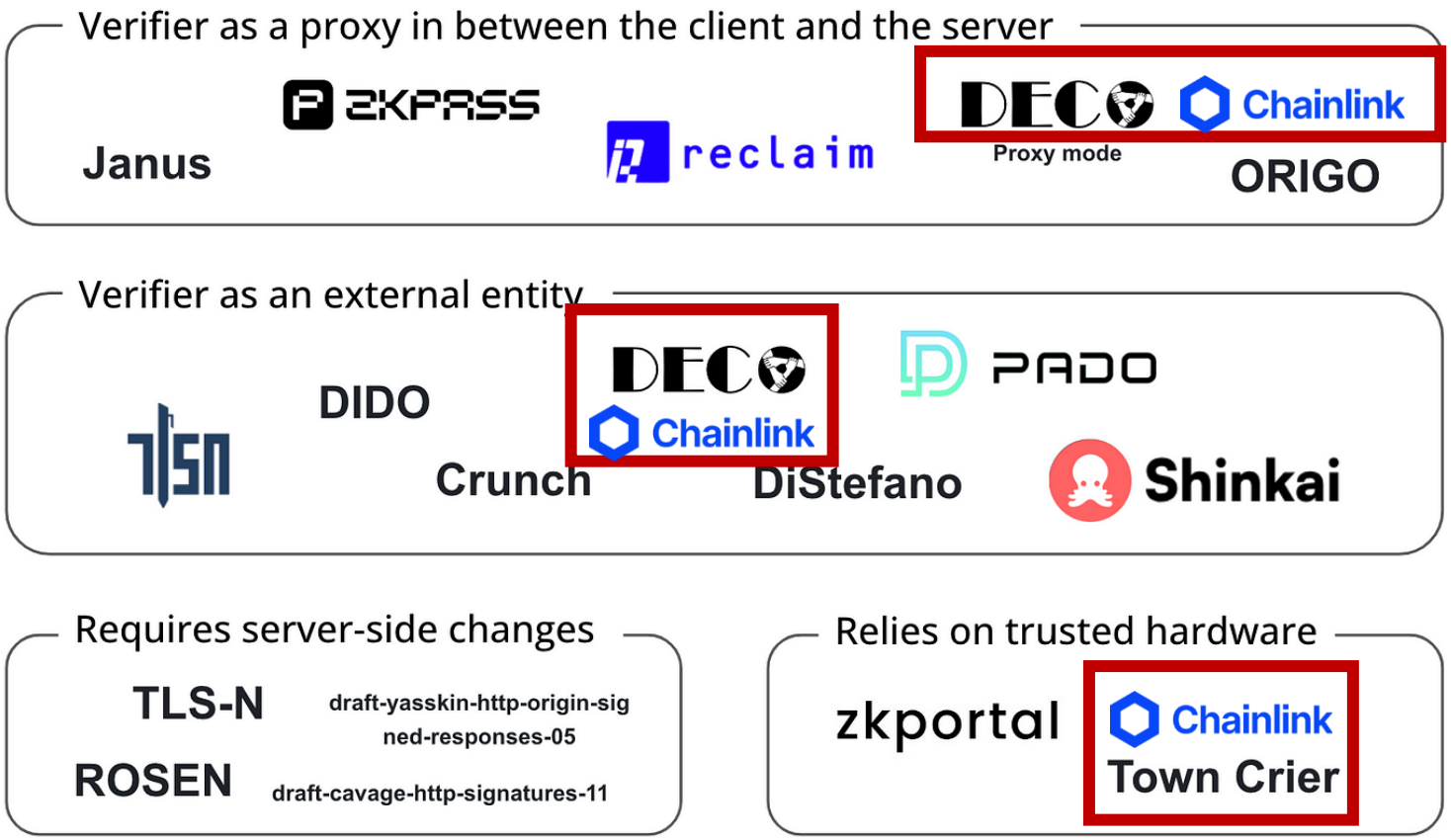
DECO

Using **cryptographic protocols**

“zkTLS” protocols

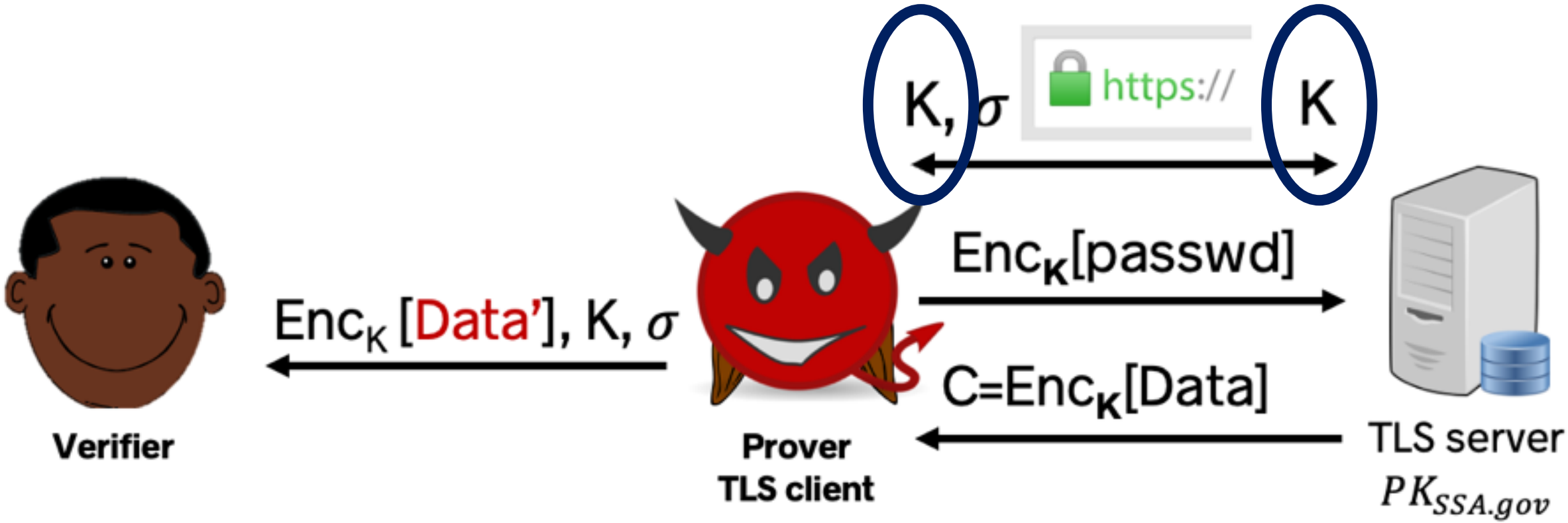
- Started by Town Crier and DECO
- “DECO” regime
  - DIDO: Data Provenance from Restricted TLS 1.3 Websites
  - Janus: Fast Privacy-Preserving Data Provenance for TLS
  - Lightweight Authentication of Web Data via Garble-Then-Prove
  - ORIGO: Proving Provenance of Sensitive Data with Constant Communication
  - DiStefano: Decentralized Infrastructure for Sharing Trusted Encrypted Facts and Nothing More
  - One final project idea:

Companies building “zkTLS” products



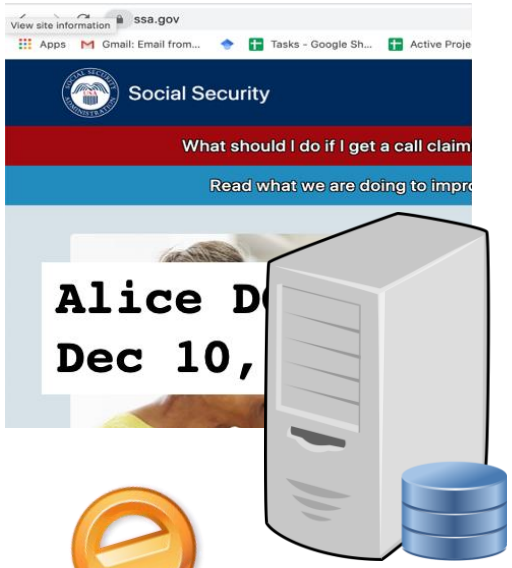
Source: https://bwetzel.medium.com/tls-oracles-liberating-private-web-data-with-cryptography-e66e5fad7c34

Problem: TLS doesn't support provenance

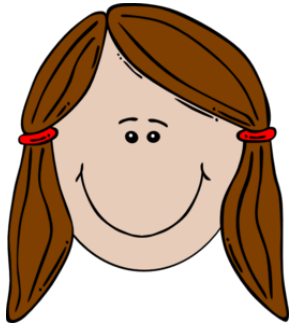


Our approach: hide session key from the prover until she commits.

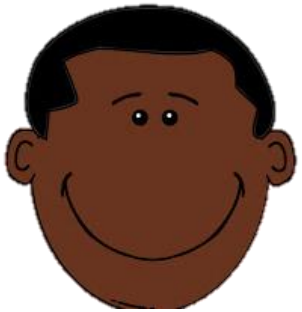
Prevent cheating by **splitting session keys**



Prover



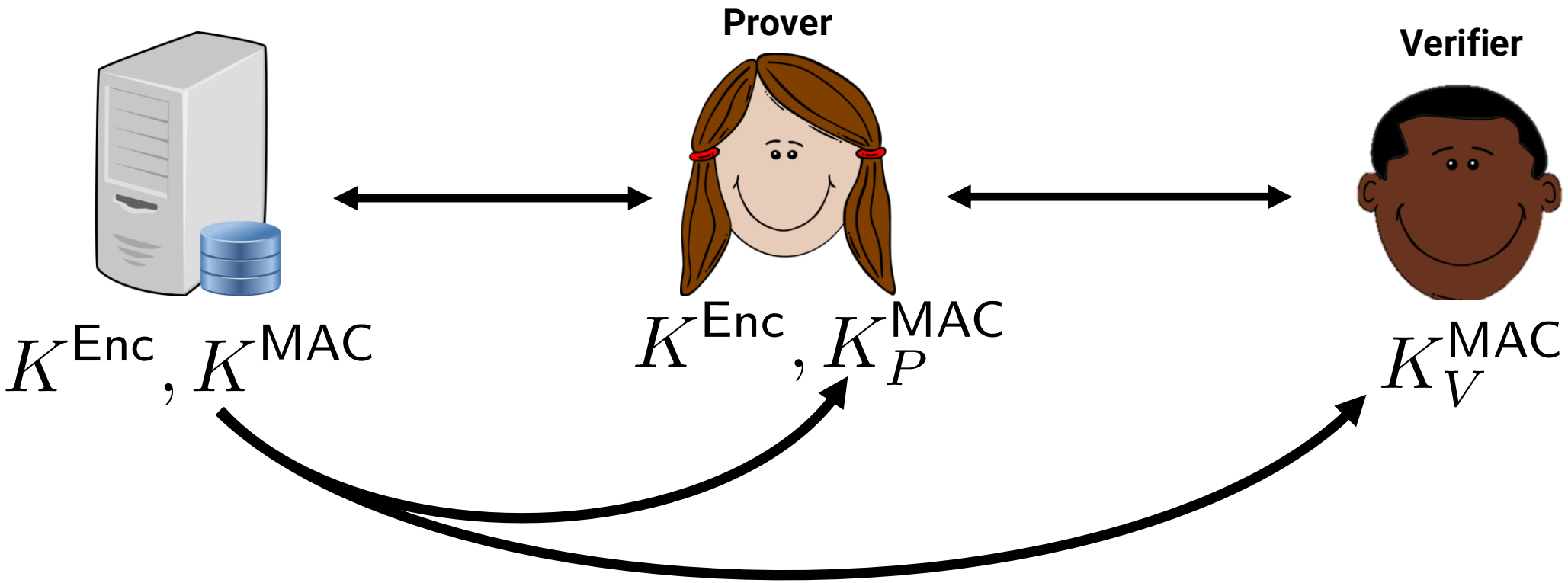
Verifier



Main idea: Three-party handshake

DECO logo

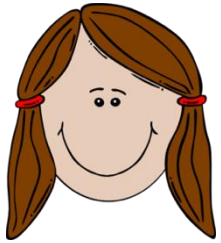
- Idea: Hide the MAC key from the prover until she commits.
- Assuming CBC-HMAC for now (GCM later)



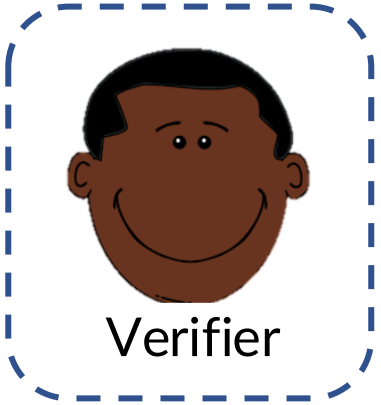
$$K_V^{\text{MAC}} \oplus K_P^{\text{MAC}} = K^{\text{MAC}}$$

Standard TLS handshake

TLS Server



TLS Client

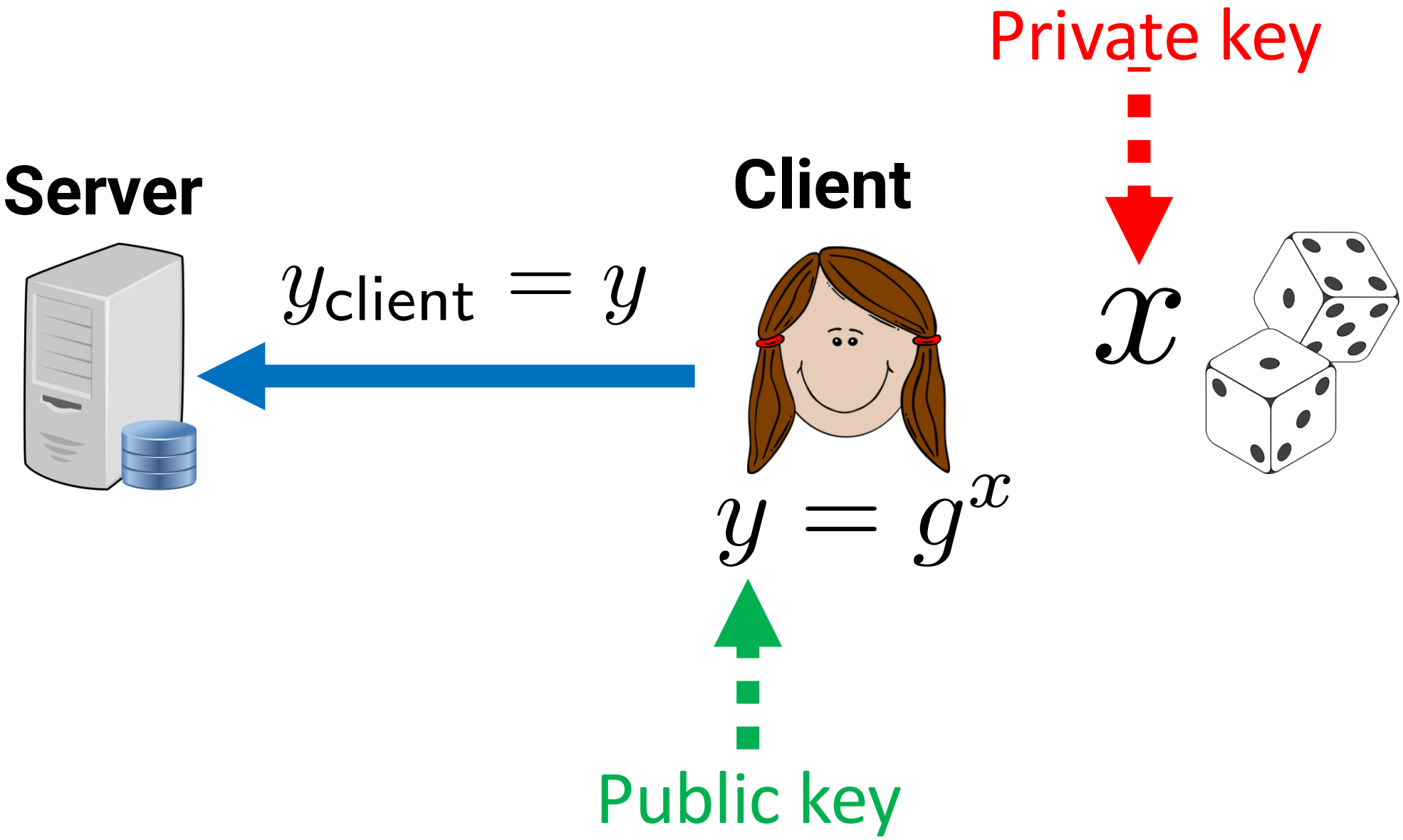


Verifier

- Key exchange (e.g. ECDHE)
- Key derivation

- Leverage the homomorphic properties of ECDHE.
- Secure Two-party Computation

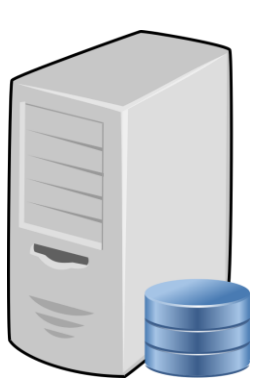
TLS: Key exchange



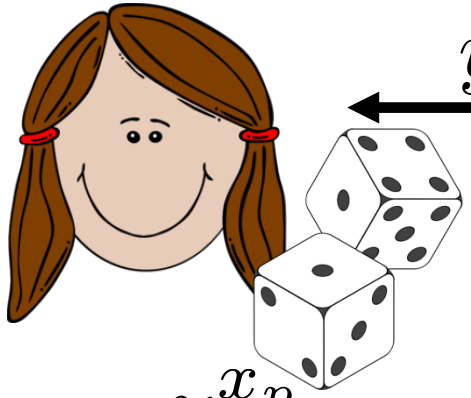
Three-party handshake: key exchange

$y_{\text{server}} = g^{x_s}$

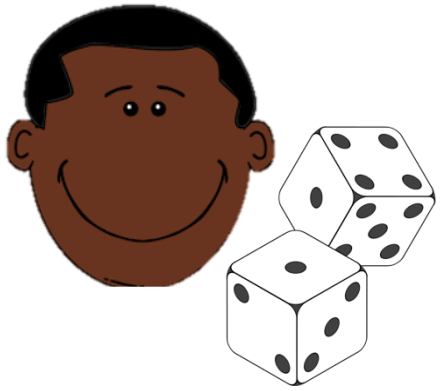
Prover **Verifier**



$y_{\text{client}} = g^{x_p} \cdot y_v$



$y_v = g^{x_v}$

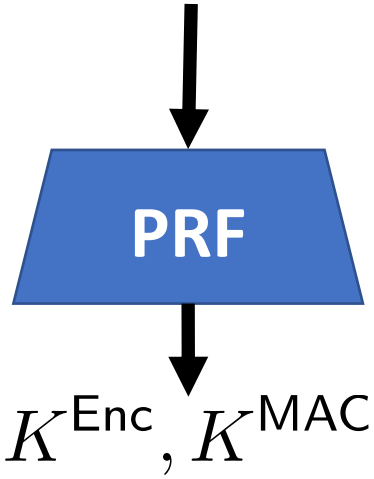


$z = y_{\text{client}}^{x_s}$ $z_p = y_{\text{server}}^{x_p}$ $z_v = y_{\text{server}}^{x_v}$

$$z = (g^{x_p})^{x_s} \cdot y_v^{x_s}$$
$$= y_{\text{server}}^{x_p} \cdot y_{\text{server}}^{x_v} = z_p \cdot z_v$$

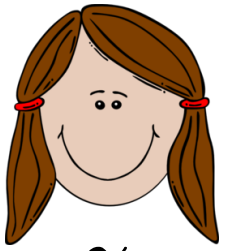
Three-party handshake: key derivation

z



$K^{\text{Enc}}, K^{\text{MAC}}$

Prover



z_p

Verifier

z_v

Can't reveal Z_P and Z_V to each other!

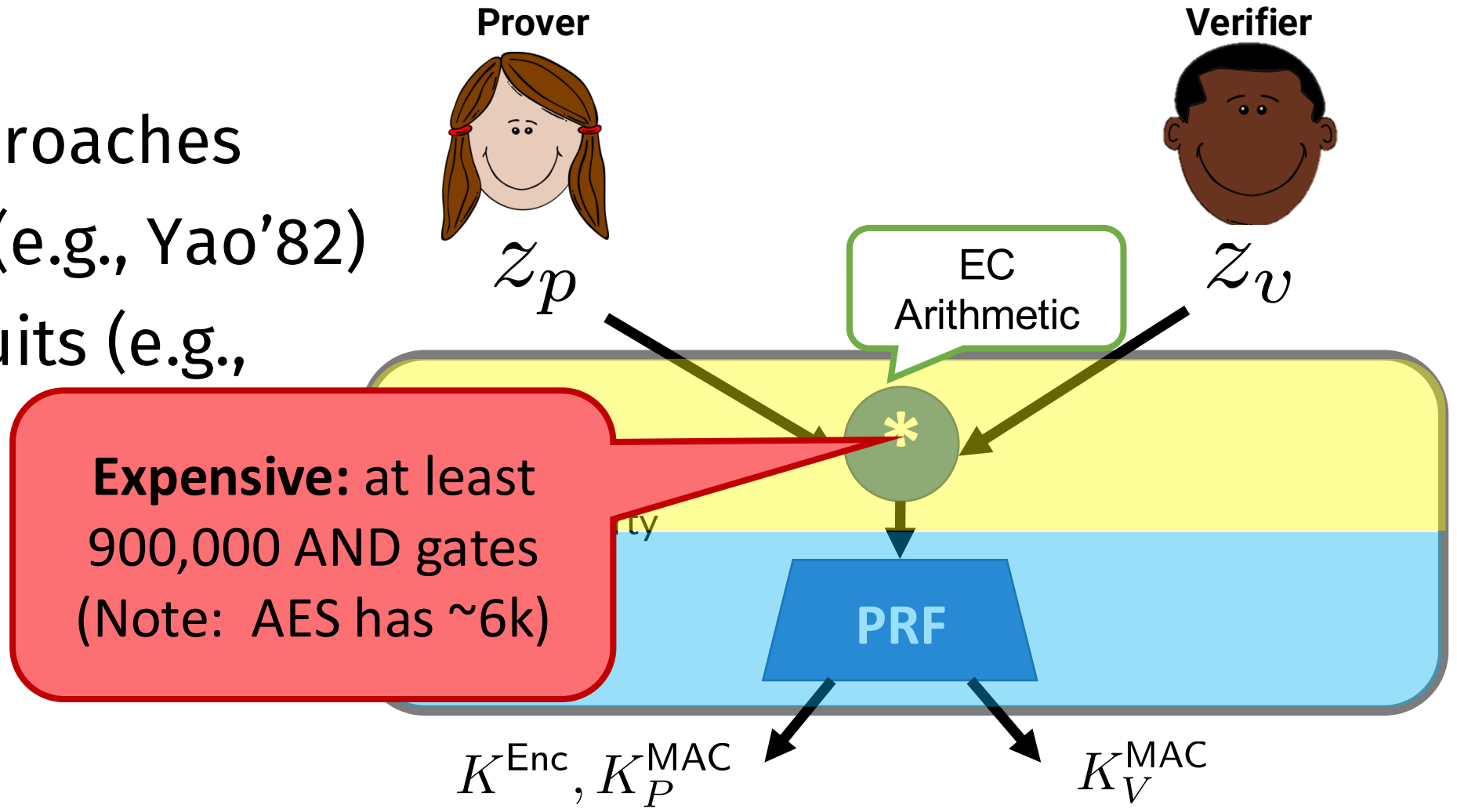
$K^{\text{Enc}}, K_P^{\text{MAC}}$

K_V^{MAC}

Secure two-party computation (2PC)

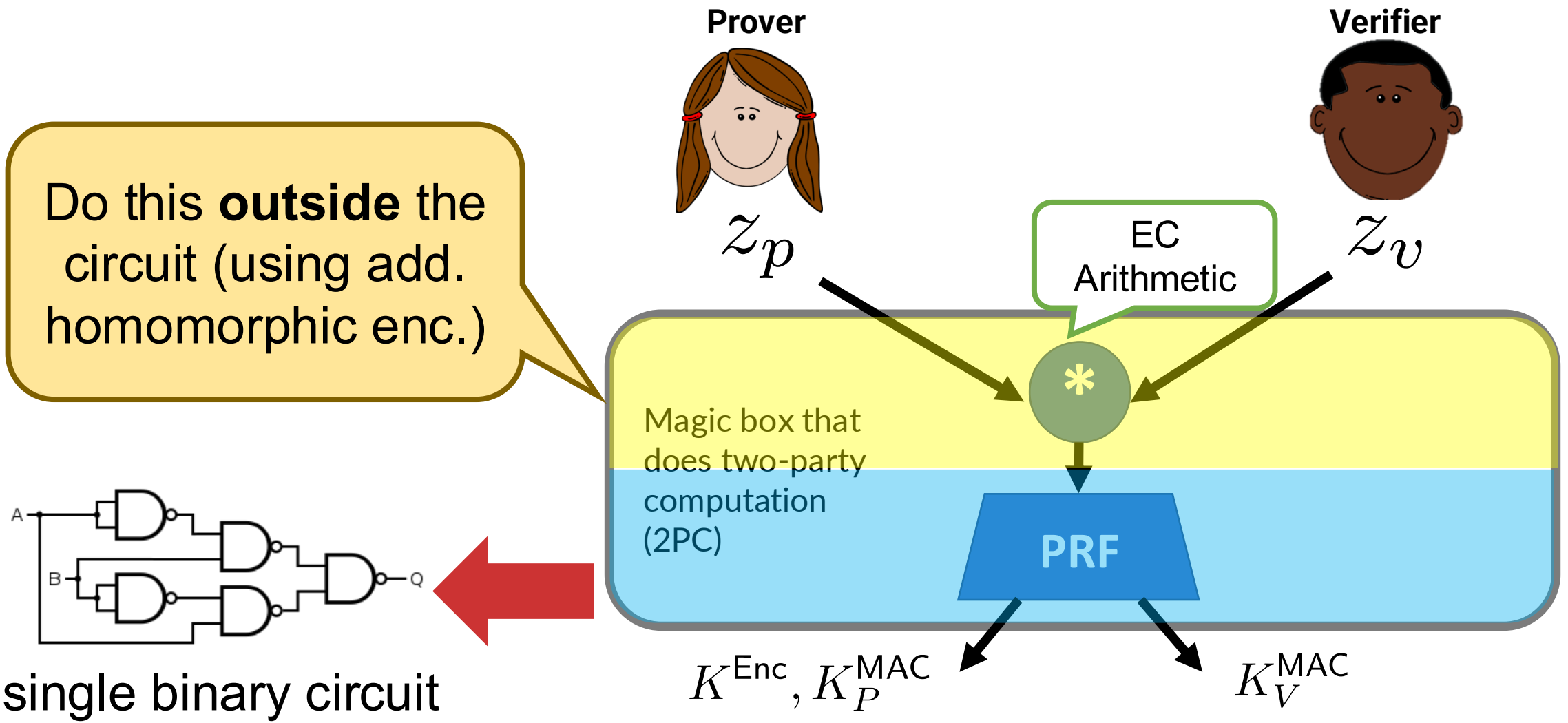
Two general approaches

- *binary* circuits (e.g., Yao'82)
- *arithmetic* circuits (e.g., BGW'88)
- but **not both**!



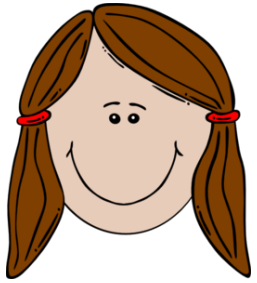
Expensive: at least 900,000 AND gates (Note: AES has ~6k)

Minimize the 2PC circuit



This denotes a TLS ciphertext.

After the handshake



Prover/TLS Client

TLS Server

K_V S->C, MAC

K_P S->C, MAC

K S->C, MAC

Compute $K=K_V + K_P$ and verify Re Response

Response K_P

Prove statements about Response

- TLS ciphertexts are commitments to plaintext data (GLR17)
- Three-party handshake makes them unforgeable

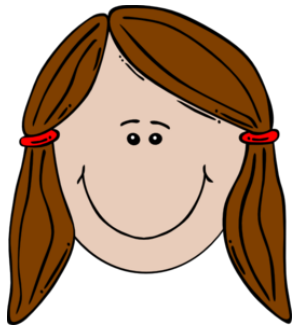
Privacy-preserving proofs

Convinced!

I am over 18

Verifier

Prover

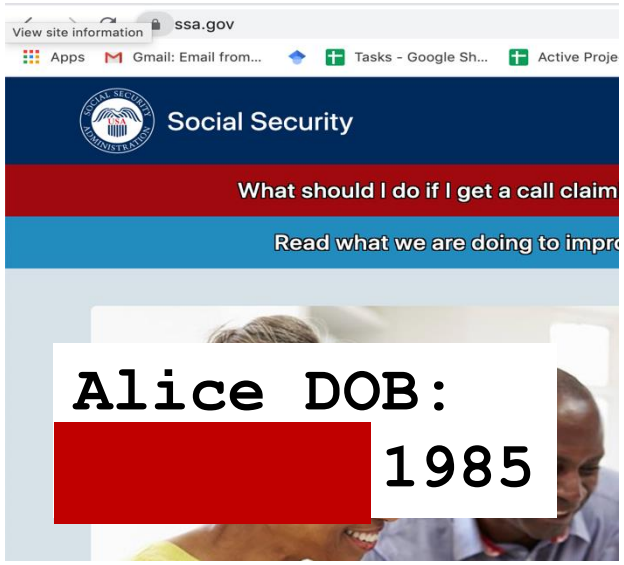


Query privacy: the verifier never sees the password.

Privacy-preserving proofs

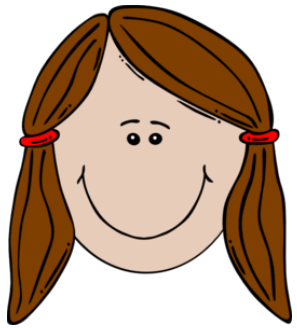
Convinced!

Verifier



I am over 18

Prover

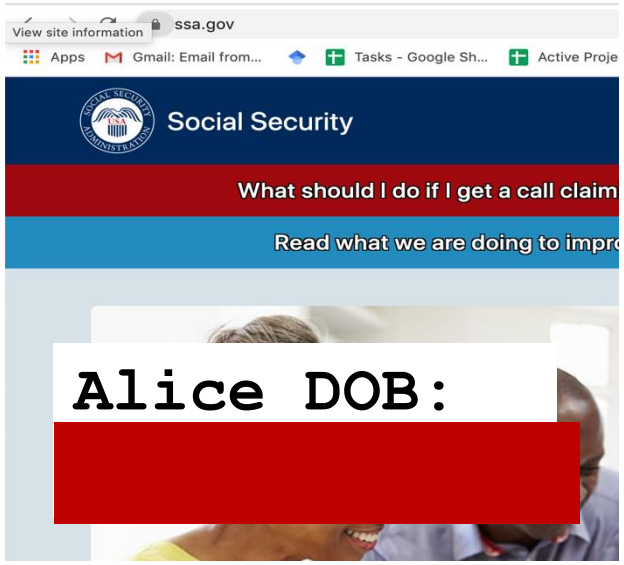


DECO supports **selective opening** for fine granular release.

Privacy-preserving proofs

Convinced!

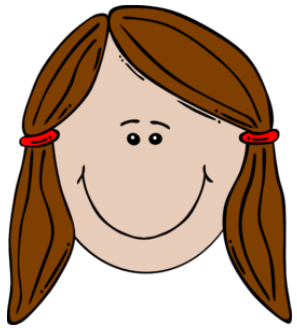
Verifier



A proof that “2020 - DOB.Year > 18”

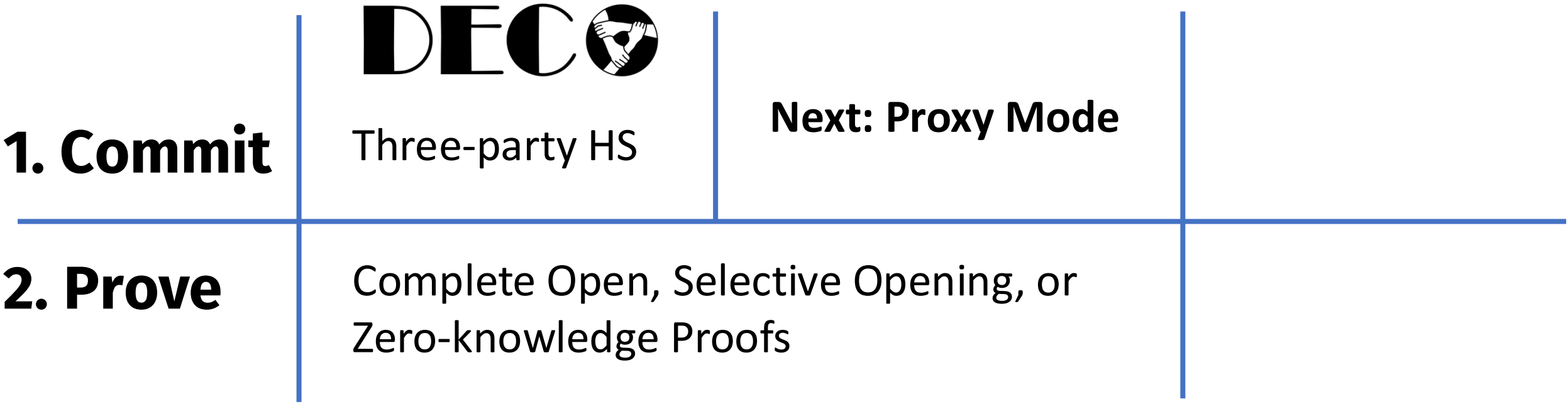
I am over 18

Prover



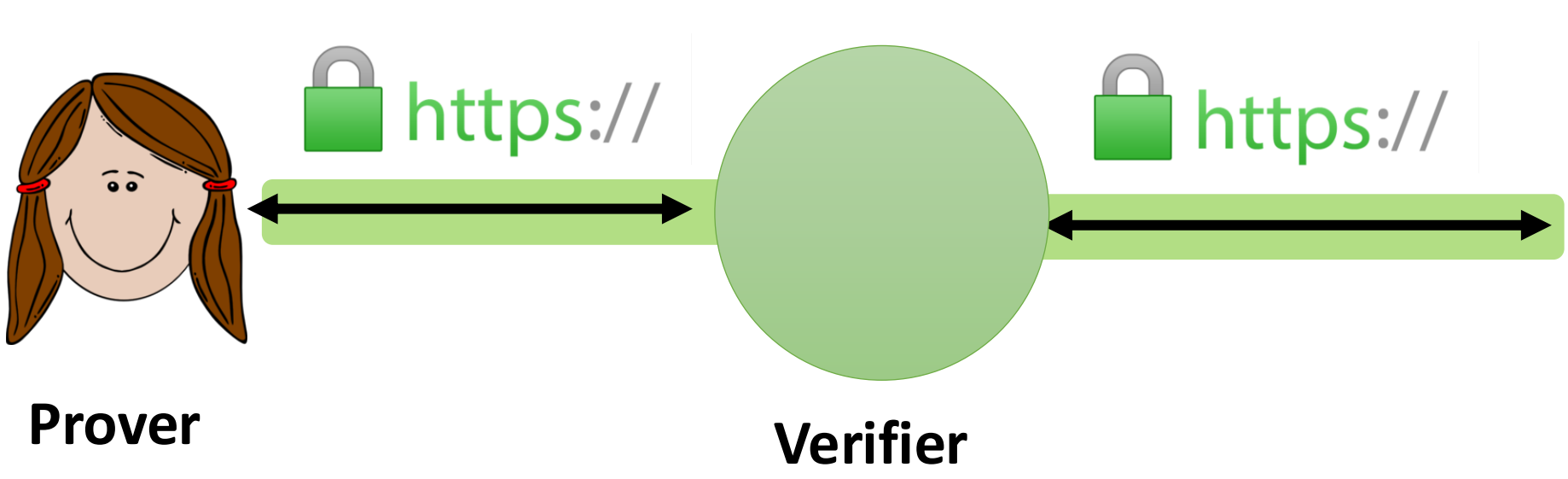
Combine selective opening with Zero-knowledge Proofs

zkTLS: proving statements about TLS data



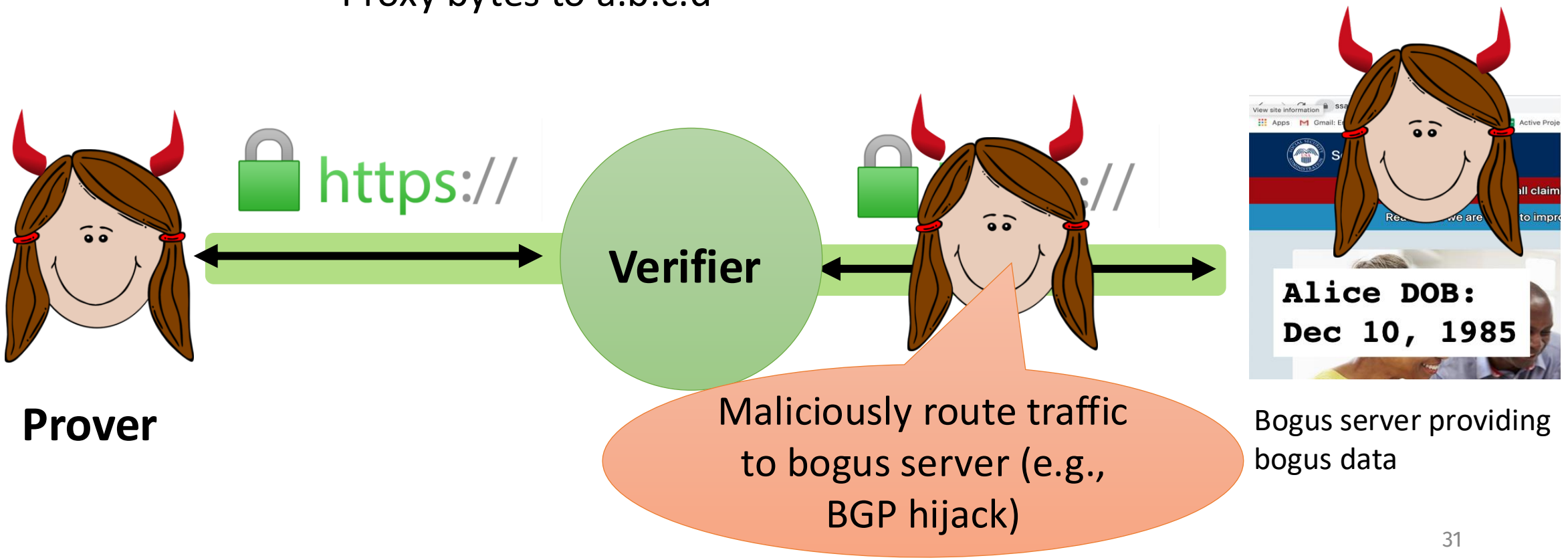
Proxy Mode: alternative to TPHS

- Idea: proxy TLS traffic through Verifier
- Verifier uses the recorded traffic as commitment to data
- Efficient alternative to third-party handshake



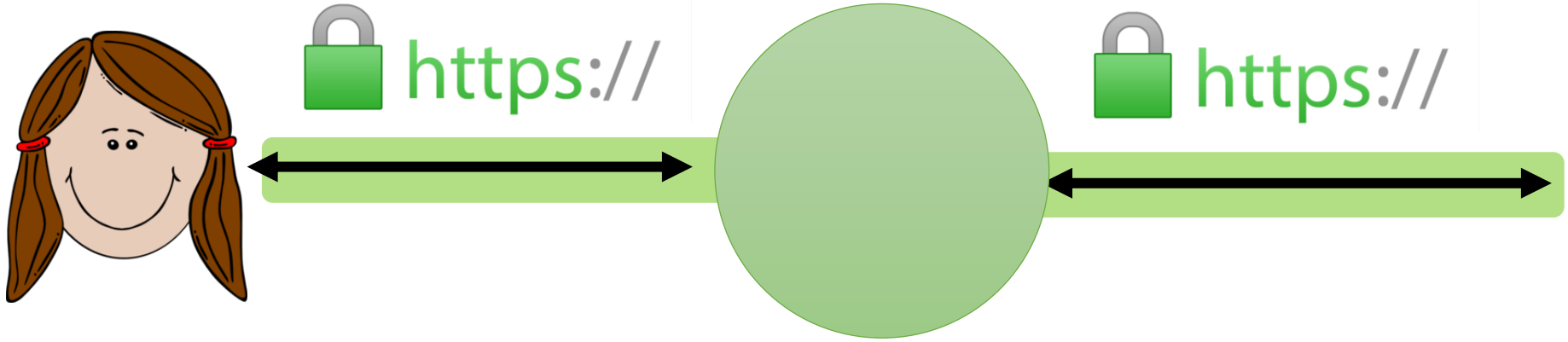
Problem #1: Relying on IP for authenticity

- Query DNS ssa.gov -> a.b.c.d (an IP addr)
- Proxy bytes to a.b.c.d



Problem #2: Does **recorded traffic** commit to the plaintext?

- Suppose the traffic encrypts message m using session key K.
- Requirement (commitment encryption): prover can't find **K'** that successfully decrypt the recorded traffic to **m'**.

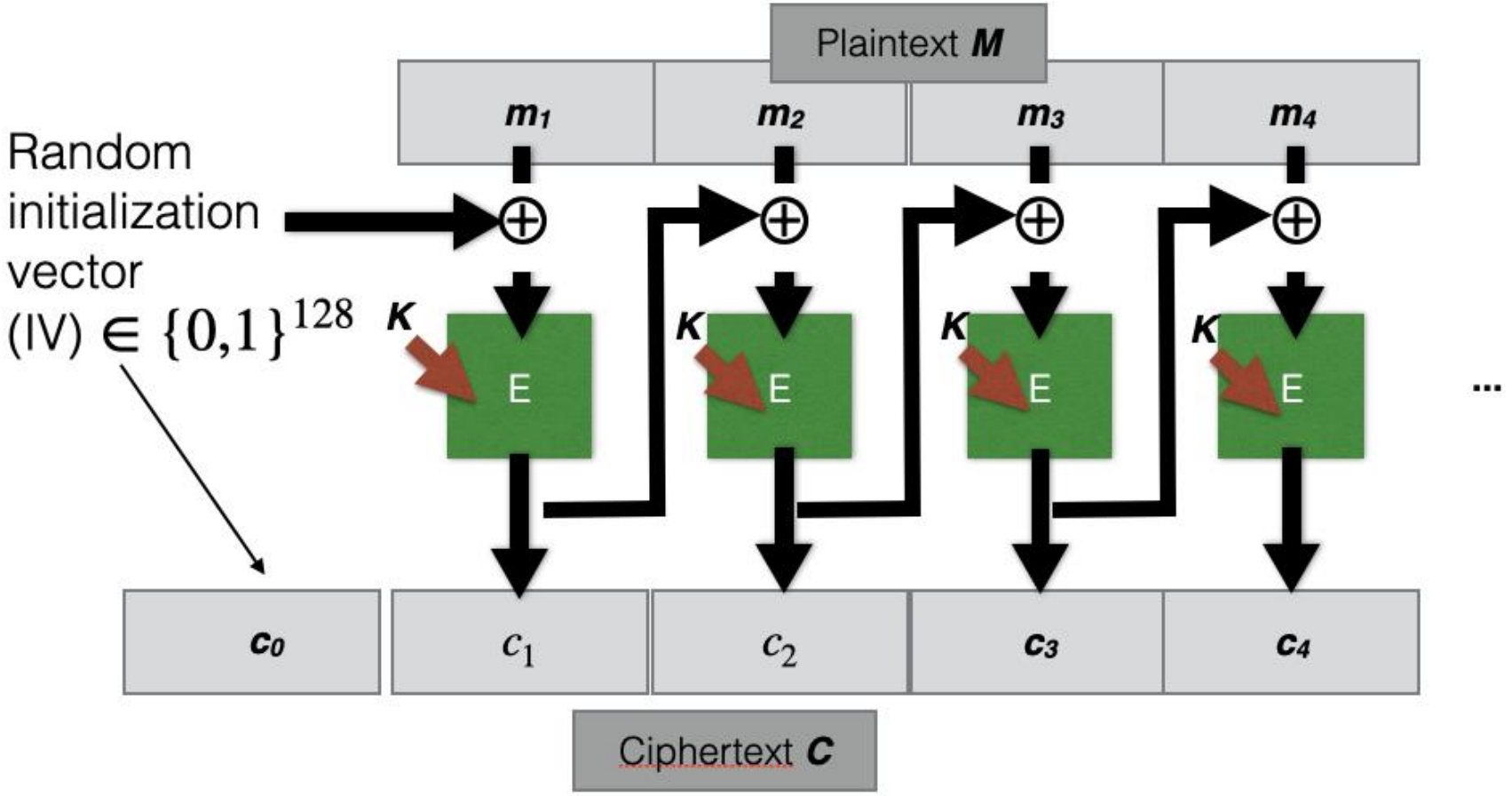


Encryption is not necessarily committing

- In general, a ciphertext $c = Enc(K, m)$ may not commit to (K, m)
- I.e., one can find (K', m') such that $Enc(K', m') = c$

- Let's see some examples

CBC is not committing: Any ciphertext can be decrypted under any key.



CTR mode is not committing (Any ciphertext can be decrypted under any key)

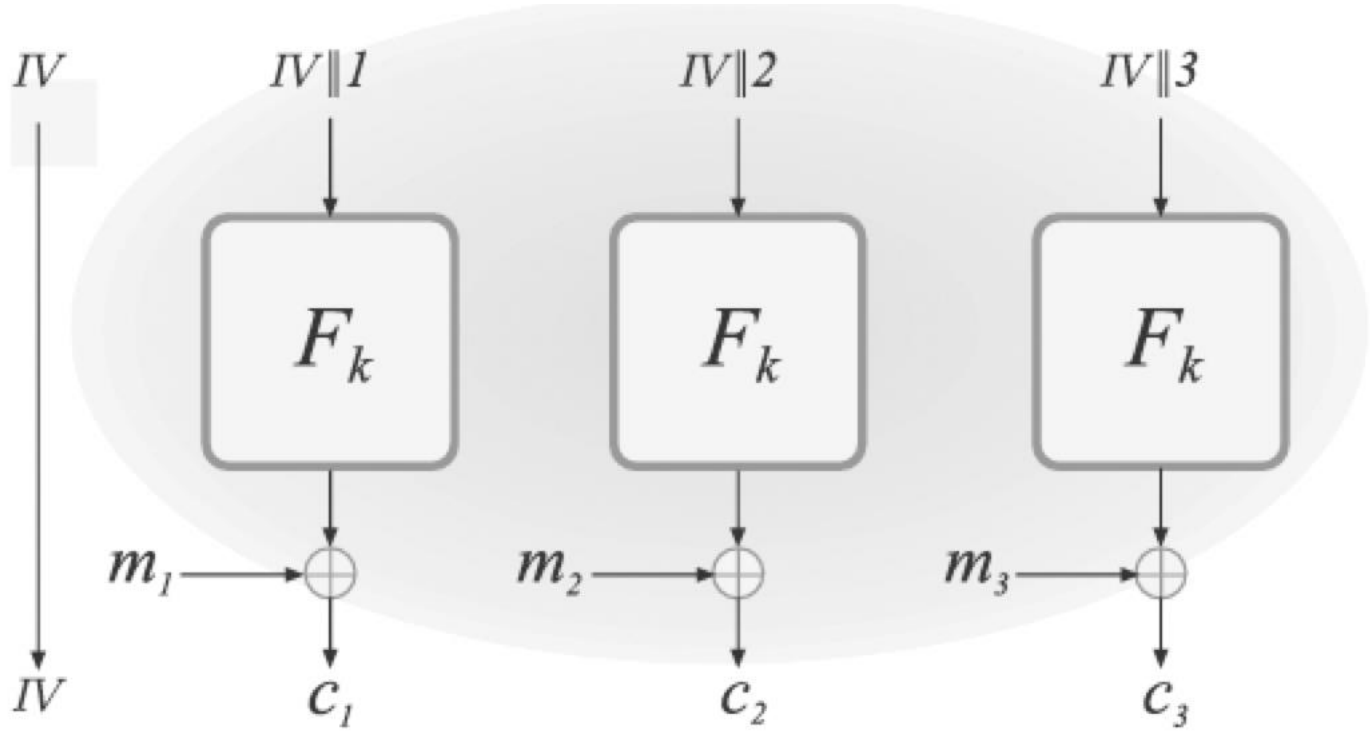


FIGURE 3.9: Counter (CTR) mode.

What about CBC-HMAC?

Enc-then-MAC(K_e, K_m, m):

- $c = Enc(K_e, m)$
- $t = Enc(K_m, c)$
- Output (c, t)

Attack:

- Output any $\hat{k}_e \neq k_e$

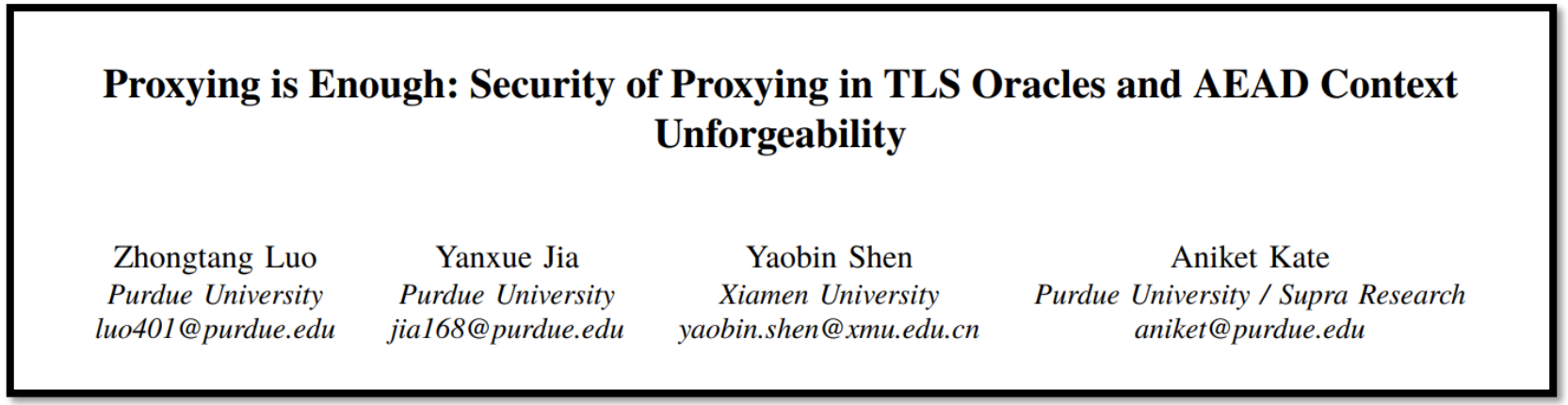
Fix: derive K_e, K_m using a KDF, which TLS does, so we are good in case of CBC-HMAC.

GCM mode is not committing

- GCM = AES-CTR + GMAC
- GMAC$(K, IV, c_1||c_2||\cdots||c_m)$:
  - $H = E(K, 0)$
  - $J = E(K, IV)$
  - Build a polynomial $P(x) = c_1 x^{m-1} + c_2 x^{m-2} + \cdots + c_m$ (in $GF(2^{128})$)
  - Return $P(H) \oplus J$
- To break commitment is to find
  - GMAC$(K, IV, c_1||c_2||\cdots||c_m)$ = GMAC$(K', IV', c_1||c_2||\cdots||c_m)$
- Attack: P is not collision resistant.

How to deal with AEAD?

- Our solution in DECO: Add key binding proofs
- Luo et al: Proxy is enough under practical assumptions
  - E.g., HTTP headers can serve as an invariant



Proxying is Enough: Security of Proxying in TLS Oracles and AEAD Context Unforgeability

Zhongtang Luo
Purdue University
luo401@purdue.edu

Yanxue Jia
Purdue University
jia168@purdue.edu

Yaobin Shen
Xiamen University
yaobin.shen@xmu.edu.cn

Aniket Kate
Purdue University / Supra Research
aniket@purdue.edu

Other Technical Subtleties

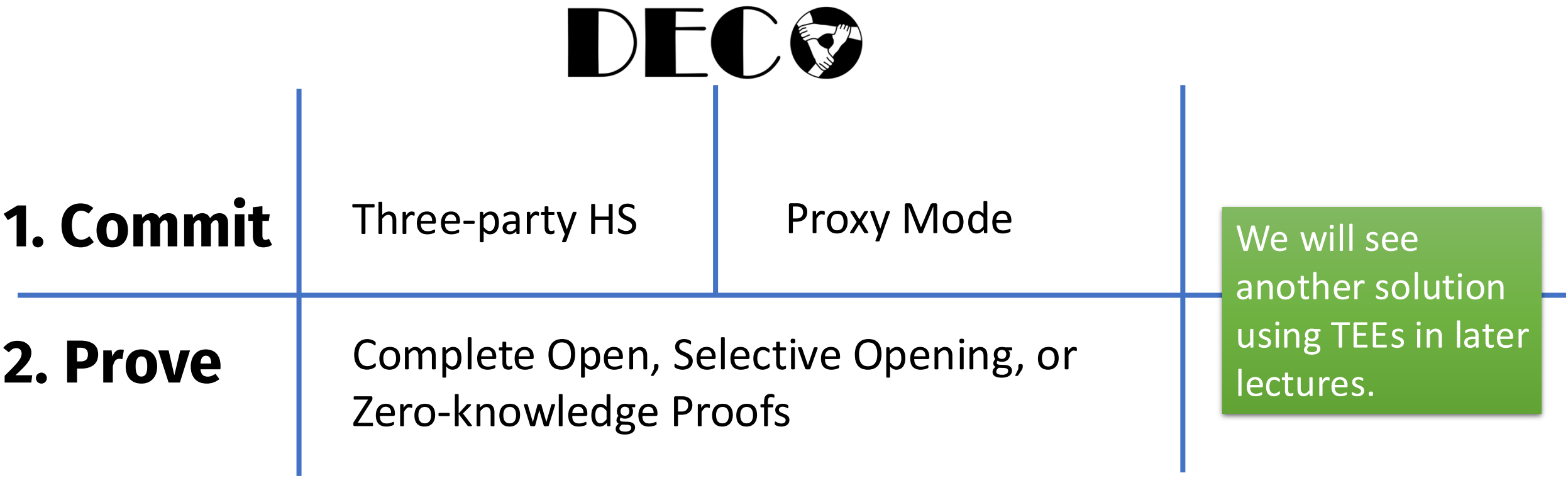
- **Context integrity** problem with privacy
  - Attackers may exploit the privacy to alter the meaning
  - Solution: careful reasoning based on the formal grammar

```
{
  “account”: ”alice”,
  ”SSN”: 1234567890,
  “balance”: 100,
  “last_month”: {“balance”: 8000},
}
```

```
{
  “account”: ”alice”,
[redacted]
[redacted]
[redacted]“balance”: 8000},
}
```

DECO addressed this problem in some special cases.
Malvai *et al.* did a thorough investigation
(https://eprint.iacr.org/2024/562).

zkTLS: proving statements about TLS data

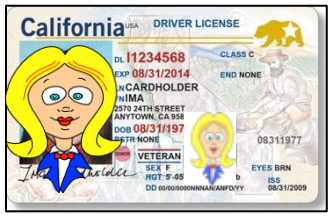


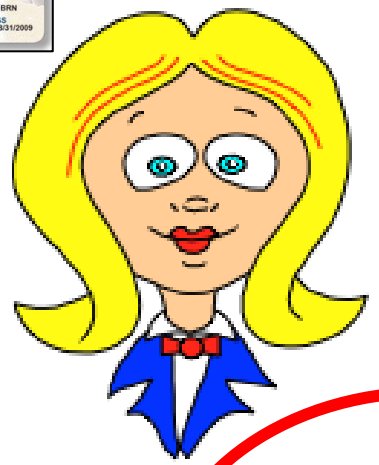
Application I: Identity

ID verification often leak more than necessary

- Bartender should only see age, not DoB, certainly not home address or immigration status.
- Mortgage approver should only see “Fan has credit > 750” not his full SSN, or his addresses in the past 10 years.
- and so on

Decentralized identity: Basic idea





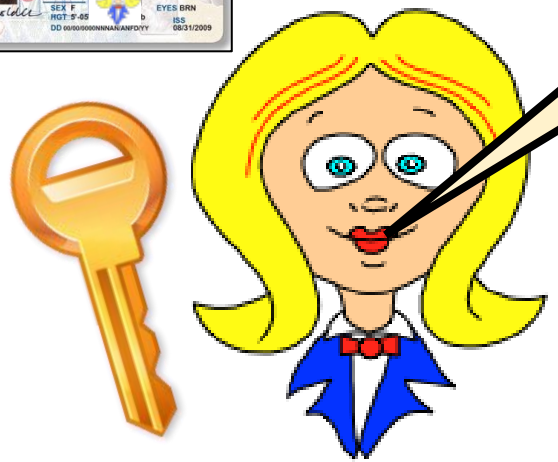
Signed credentials by DMV (hypothetically)

DMV

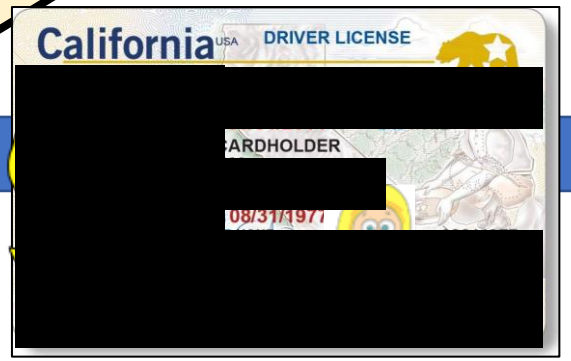
E.g., The public key can be your Ethereum Naming Services (ENS) identity

Presentation=

I'm Alice.
I'm over 18.
I'm a CA resident.



SK

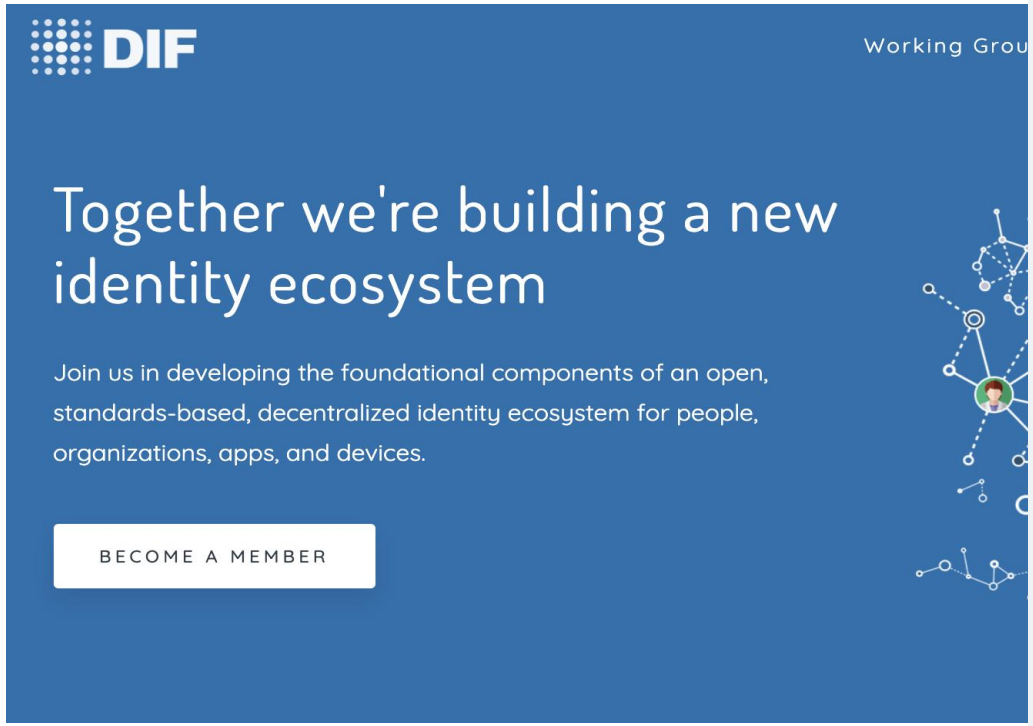


We will discuss the protocols for DID in later lectures.

Verify( PK , 

PK, claims)

DIDs in the work

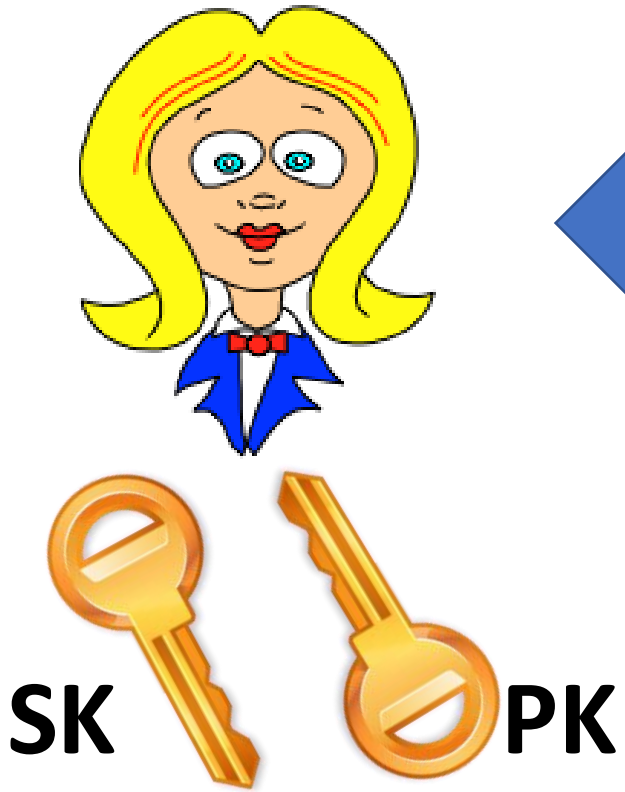


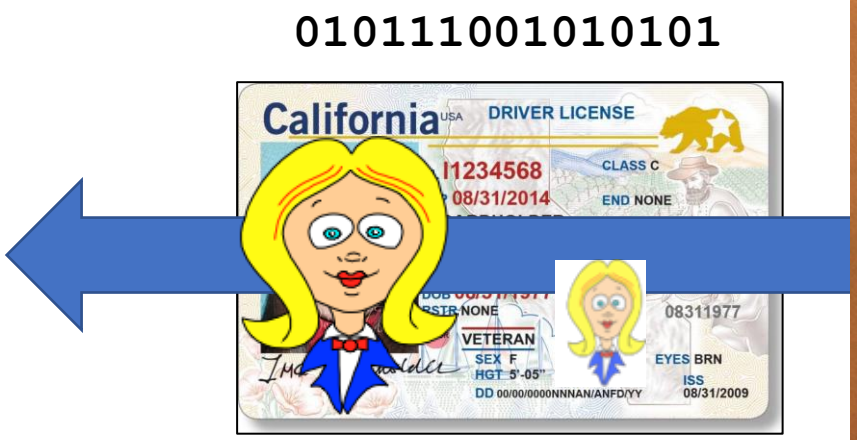
Decentralized Identifiers (DIDs) v1.0

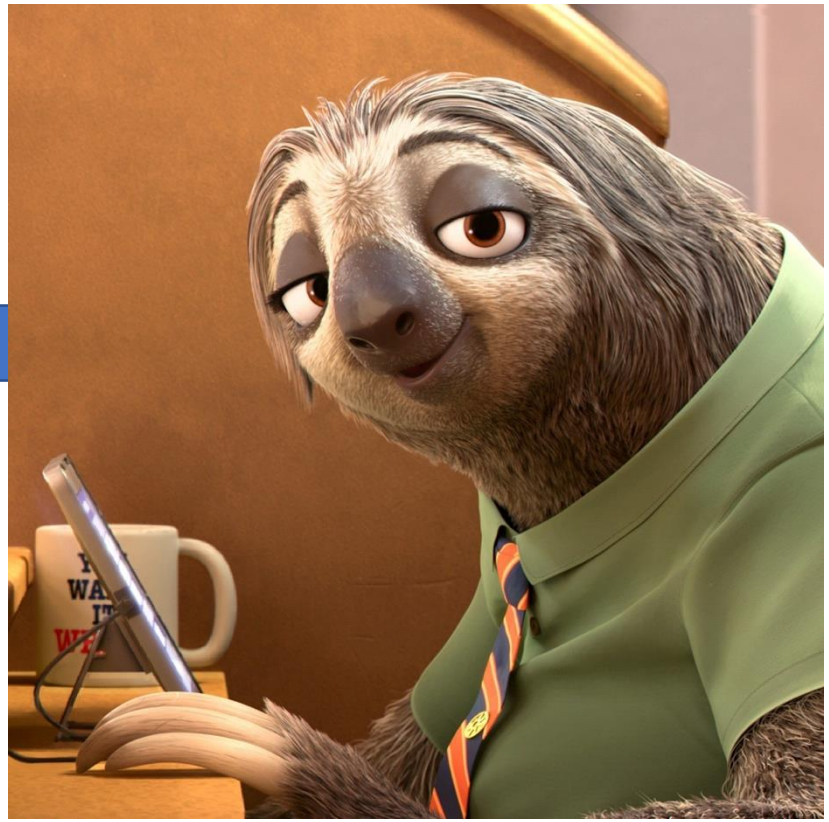
Core architecture, data model, and representations

W3C Recommendation 19 July 2022

Bootstrap challenges: **Guess when will DMV start doing this?**

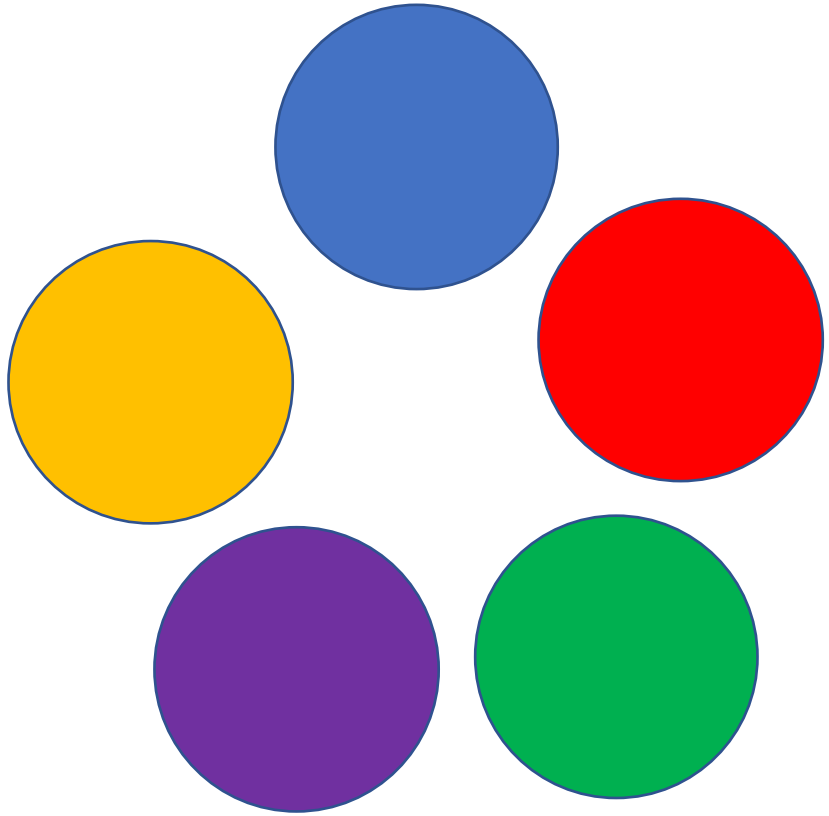






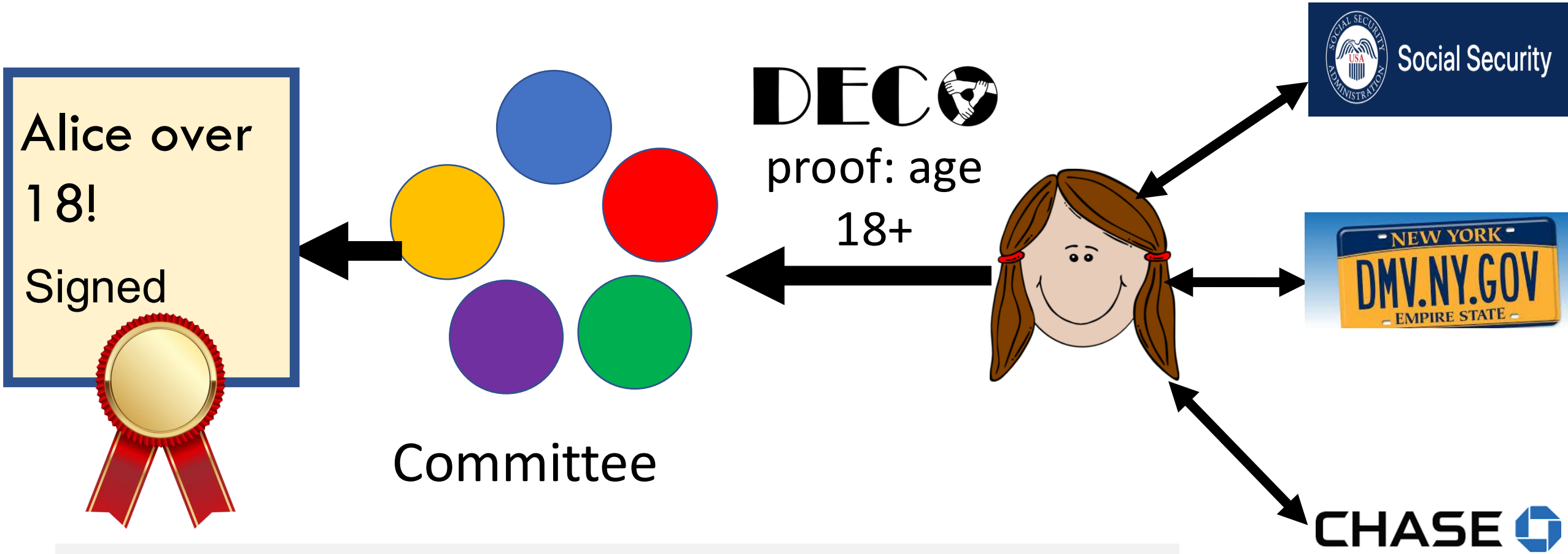
Basic setup: The committee

- *k*-out-of-*n* honest
- Securely store *secrets*
  - Via secret sharing [Shamir ‘79]
- Securely store identity data
- Verify proofs



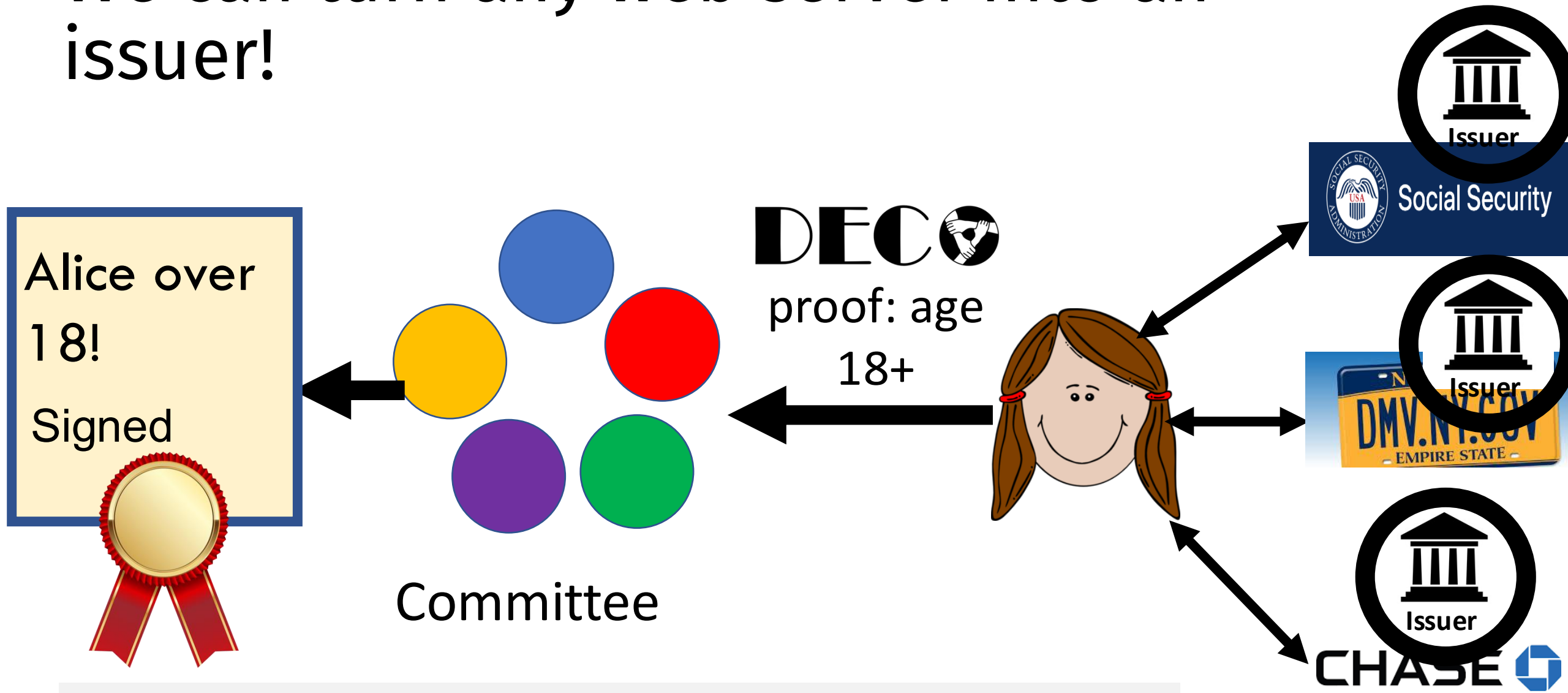
“CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability” Maram et al, In IEEE S&P 2021.

Credential issuance



"CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability" Maram et al, In IEEE S&P 2021.

We can turn any web server into an issuer!

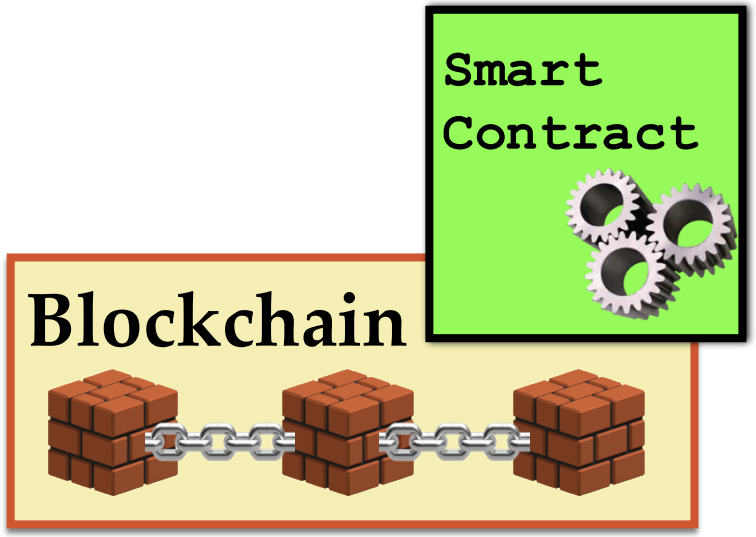


"CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability" Maram et al, In IEEE S&P 2021.

Application II:
Smart Contract Oracles

Smart contracts are a big step forward

- **Ethereum** launched in 2014
- **Smart contracts**: programs running on a blockchain.



- ✓ Integrity
- ✓ Transparency
- ✓ Availability

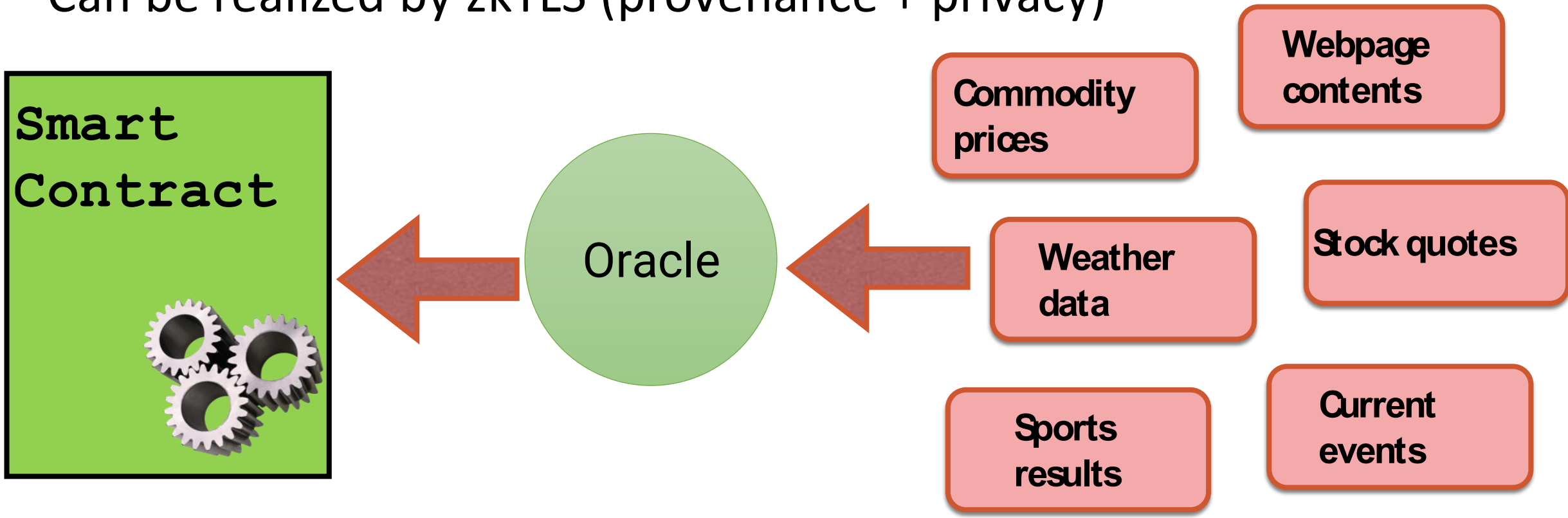
Not so smart: Smart contracts can't access data about the real world

- What applications need real-world data?
- **Pretty much anything interesting**: stablecoins, prediction markets, Synthetic assets, supply chain tracking, etc.

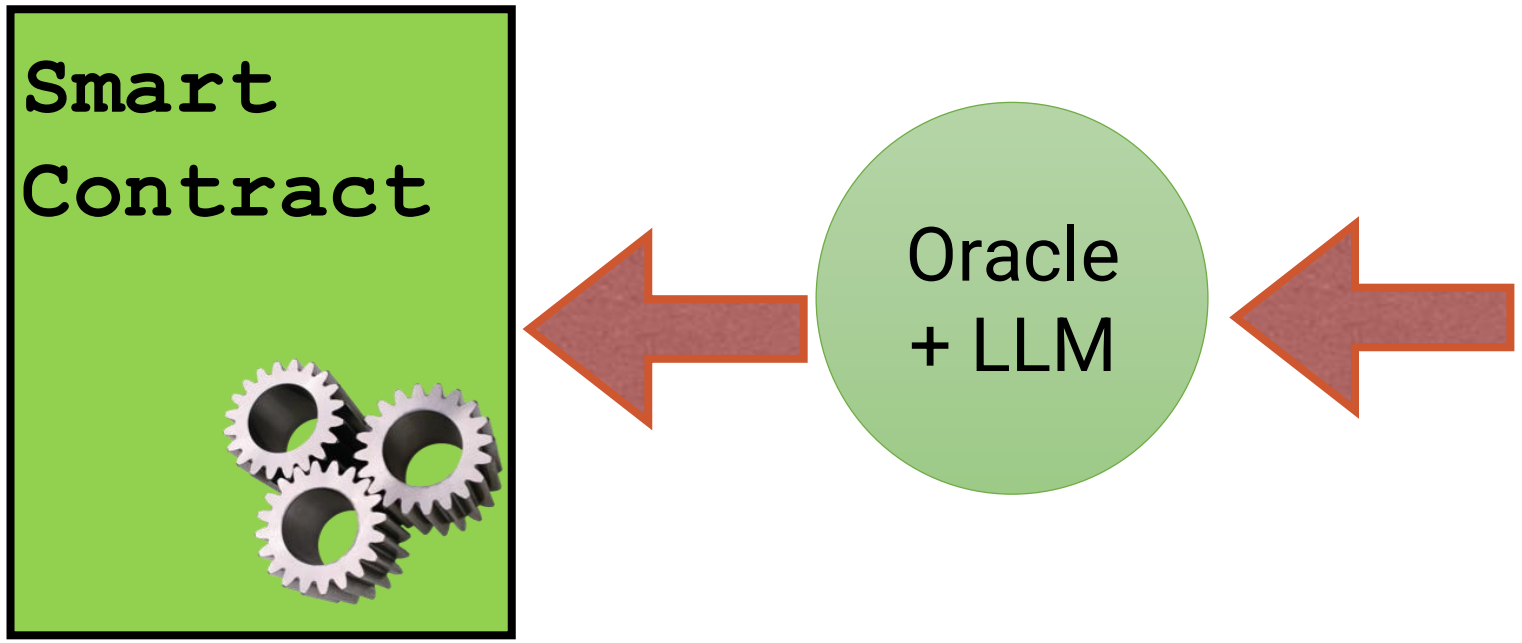
Data about the real world

Solution: Oracles

Can be realized by zkTLS (provenance + privacy)



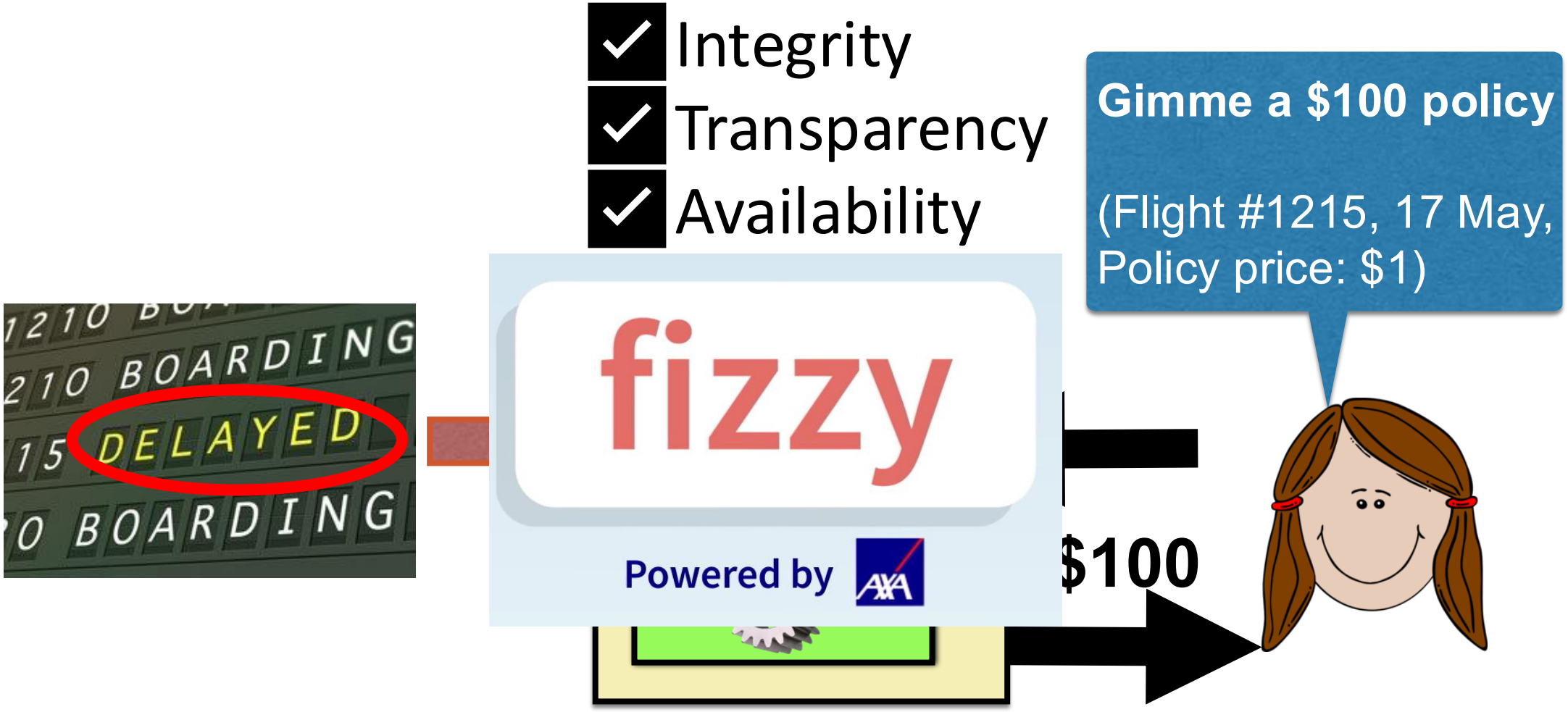
An opportunity to make smart contract **much smarter**



- Legal documents
- market reports
- news articles
- poll results
- **pretty much anything digital**

To learn more: “Empirical Evidence in AI Oracle Development” by Chainlink Labs.

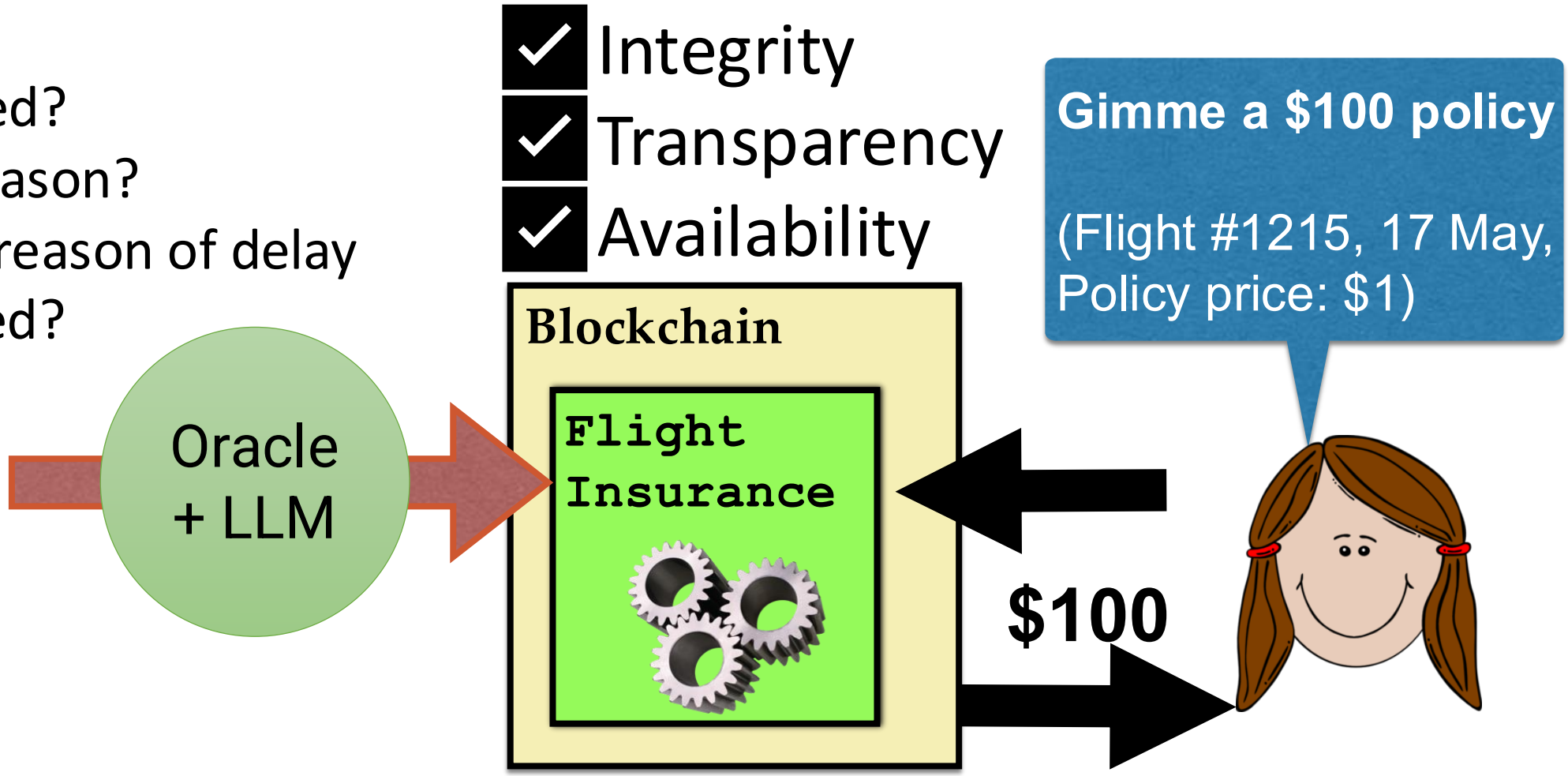
Cool Example: flight delay insurance



- Limitation: only support simple insurance terms.

Cool Example: flight delay insurance

- Delayed?
- The reason?
- Is the reason of delay covered?



Application III: AI

Proving provenance of training data

Props for Machine-Learning Security

Ari Juels[1] and Farinaz Koushanfar[2]

[1]Cornell Tech
[2]University of California San Diego

October 29, 2024

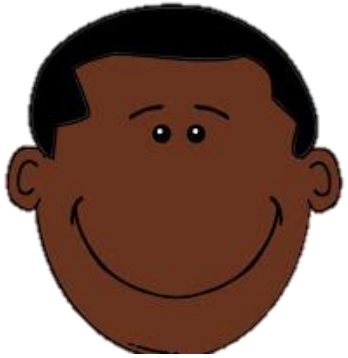
Abstract

We propose *protected pipelines* or *props* for short, a new approach for authenticated, privacy-preserving access to deep-web data for machine learning (ML). By permitting secure use of vast sources of deep-web data, props address the systemic bottleneck of limited high-quality training data in ML development. Props also enable privacy-preserving and trustworthy forms of inference, allowing for safe use of sensitive data in ML applications. Finally, props offer a new approach to constraining adversarial inputs. Props are practically realizable today by leveraging privacy-preserving oracle systems initially developed for blockchain applications.

Many nice ideas on oracle's AI applications by Juels and Koushanfar (https://arxiv.org/pdf/2410.20522).

Proving provenance of AI outputs

Verifier

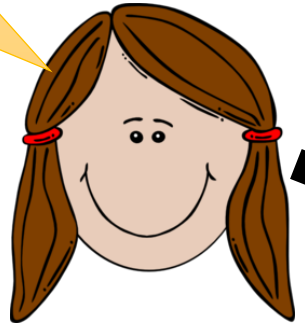


Algorithm Audit

E.g., researchers collecting data

Here are my YouTube recommendations

Prover



Login: password

Researcher's goals:

- Verify data correctness
- Respect user privacy

Summary: oracles

DECO

- Originated as systems to supply verifiable data to smart contracts, but their applications extend to **digital identity, social media, and AI.**
- Town Crier and DECO were among *the first* to formalize oracle security and realize it via verifiable provenance of TLS-encrypted data, turning HTTPS websites into sources of verifiable claims.
- These works initiated a new line of research and many real-world implementations, now known as **zkTLS**.