Public Key Infrastructure (PKI)
Tutorial for CANS'20
Day 3: CA Failures and Certificate Transparency

Amir Herzberg
University of Connecticut

See ch. 8 of 'Applied Intro to Cryptography',
available at my site: https://sites.google.com/site/amirherzberg/home.

PKI Tutorial – CANS’20: Agenda

- **Day 1: Introduction, X.509 and constraints**
- **Day 2: Revocations and Merkle Digests**
- **Day 3: CA failures**
  and Certificate Transparency
- Conclusions, directions and challenges

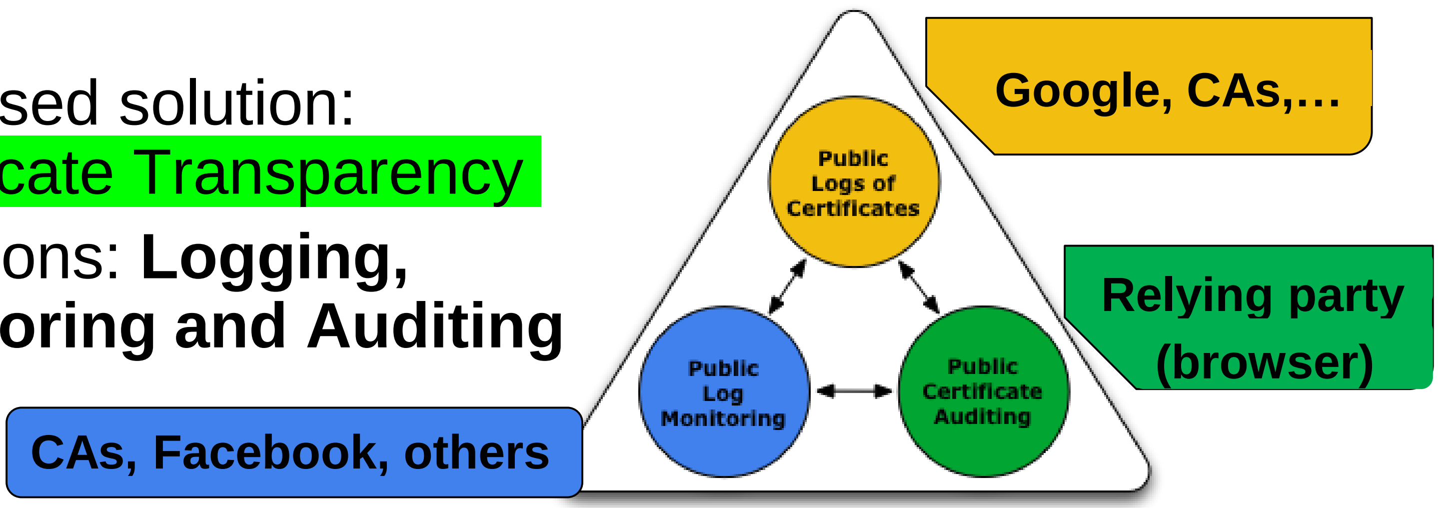
Defenses against CA failures

- **Use name constraints to limit risk**
  - But… which CA(s) will 'own' global TLDs (.com, etc.)?
- **Static key pinning:** 'burned-in' public keys
  - Detected MitM in Iran: rogue DigiNotar cert of Google
  - Limited: changing keys? Which keys to preload ?
- **Dynamic Pinning: HTTP Public-Key Pinning (HPKP)**
  - Server: I always use this PK / Cert / Chain
  - Client: remember, implement, detect & report attacks
  - Concerns: key loss/exposure, changing keys (recover security)
  - CA-pinning may work better
- Certificate Transparency (CT): **real accountability !**
  - **Public, auditable certificates log**

Certificate Transparency (CT) [RFC6962]

- X.509, PKIX: CAs sign cert
  - **Accountability**: identify issuer, **given** (rogue) cert
- Challenge: find rogue cert
  - Unrealistic to expect relying parties to detect !
  - Google detected in Iran - since Chrome had pinned Google's PK
- Proposed solution: Certificate Transparency
- Functions: **Logging, Monitoring and Auditing**

- **Loggers** provide public logs of certificates
- **Monitors** monitor certificates logged for detection of suspect certificates
  - And detect bad loggers ?
- **Auditing** (auditors?): check for misbehaving loggers



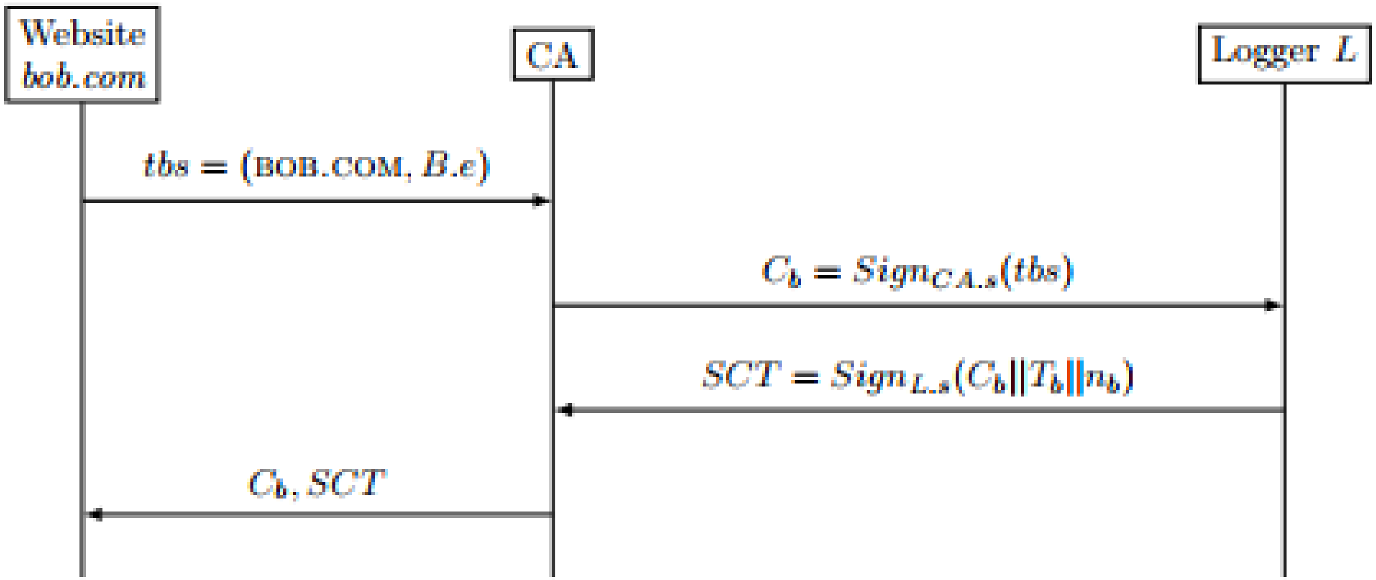
Certificate Transparency (CT): Goals

- ➔ Easier to detect, revoke rogue certificates
- ➔ Easier to detect, dis-trust rogue CAs:
  No (real) accountability without transparency !
- What about rogue loggers, monitors ?
- Option 1: Honest-Logger CT (HL-CT)
  - Assume honest logger [or out of two loggers – redundancy; ~ Chrome]
- Option 2: AnG-CT: Audit and Gossip to detect rogue logger
- Option 3: No Trusted Third Party (NTTP-Secure CT)
  - Monitors, relying-parties detect misbehaving loggers
  - Relying party decides which **monitor(s)** to rely on (trust) !
  - Original CT goal

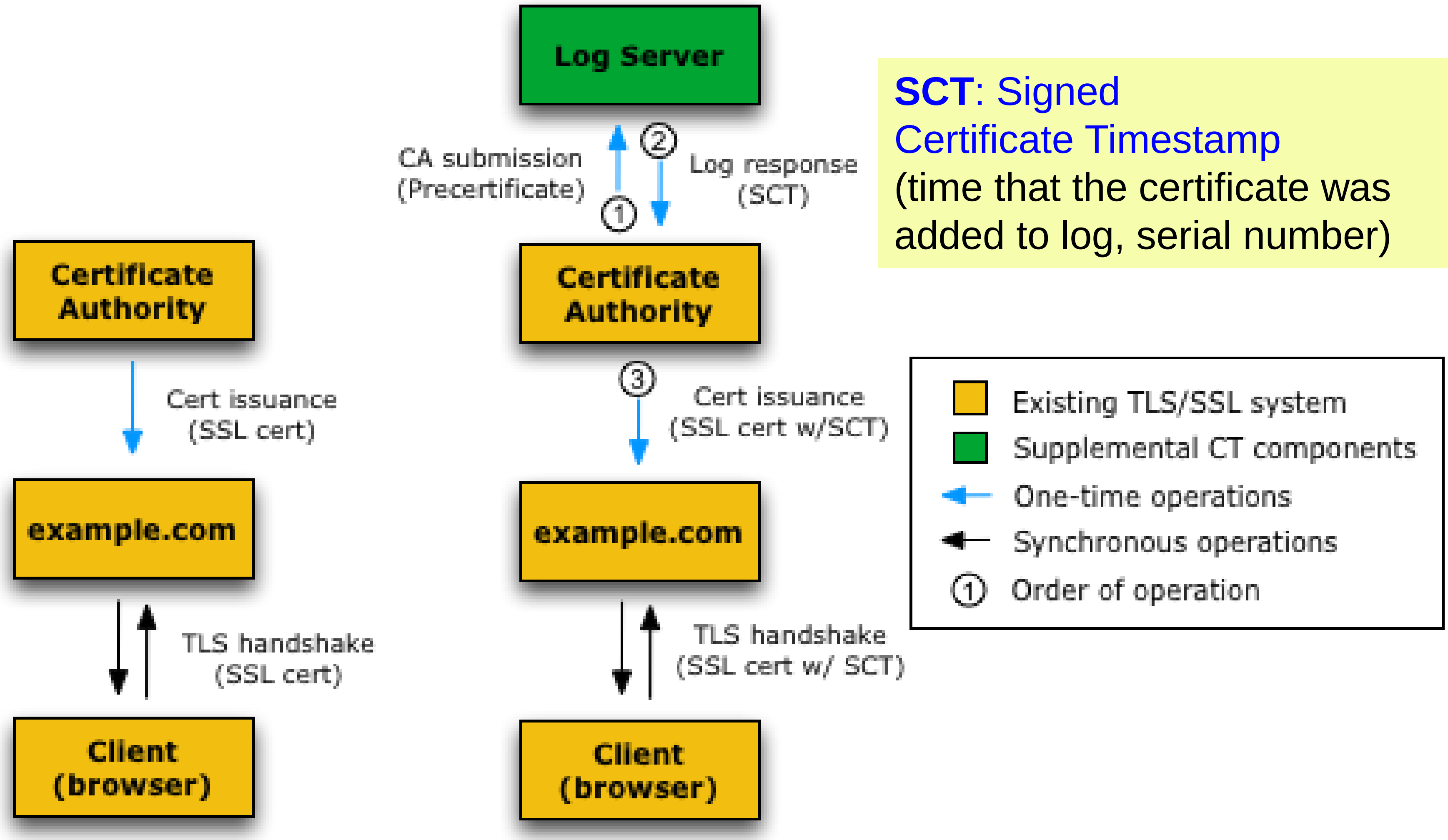
Honest-Logger CT: Issuing Certificate

- Subject, e.g. website, sends request
  - Request contains 'To Be Signed' fields: name, public-key
- CA validates request, signs cert, sends to **logger**
- Logger adds cert to log, signs and returns (signed) **SCT**
- CA sends cert + SCT to subject (e.g., website)

TBS=To Be Signed



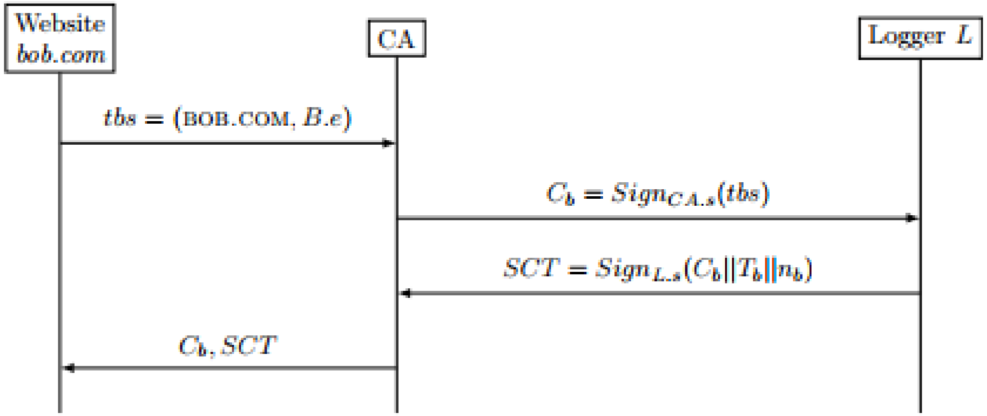
X.509 vs. HL-CT: Issuing process



SCT: Signed Certificate Timestamp
(time that the certificate was added to log, serial number)

Honest-Logger CT: Issuing Certificate

- Issuer (CA) must send every cert to logger
- Logger returns Signed Certificate Timestamp (SCT)
  - Validate that the cert was logged at given time
- CA gives cert, SCT to subject (e.g., website)
- Subject sends SCT (with cert) to relying party
- Relying party 'knows' cert was logged (and when)
- How do we use logs to detect rogue certs?



Detecting rogue certs in log: Monitors

Goal: early detection of rogue certs in log

Logs should be publicly available

Name-owners can monitor the log

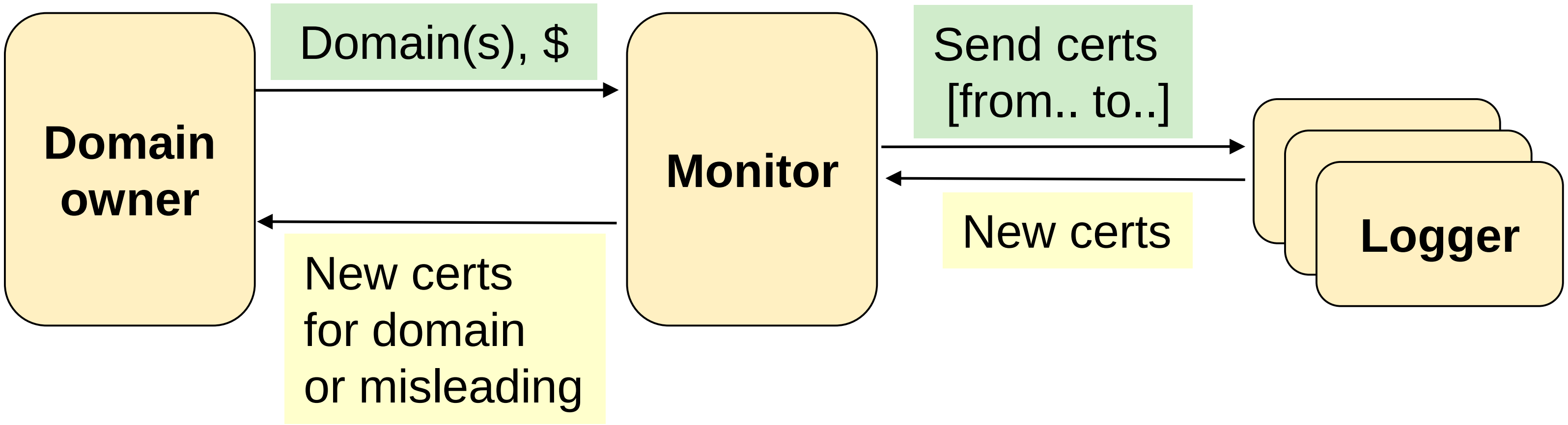
- Download, check log for relevant names
- ☹ high overhead to everyone!

Instead: monitors do this (for many names)

- Several such monitors, loggers already operate
- Download only new certificates
  - And: ask log for seq# and/or date of last logged cert
  - Ask log to send range of certs: <from-to>
  - Optionally: maintain all certs (to check new names)

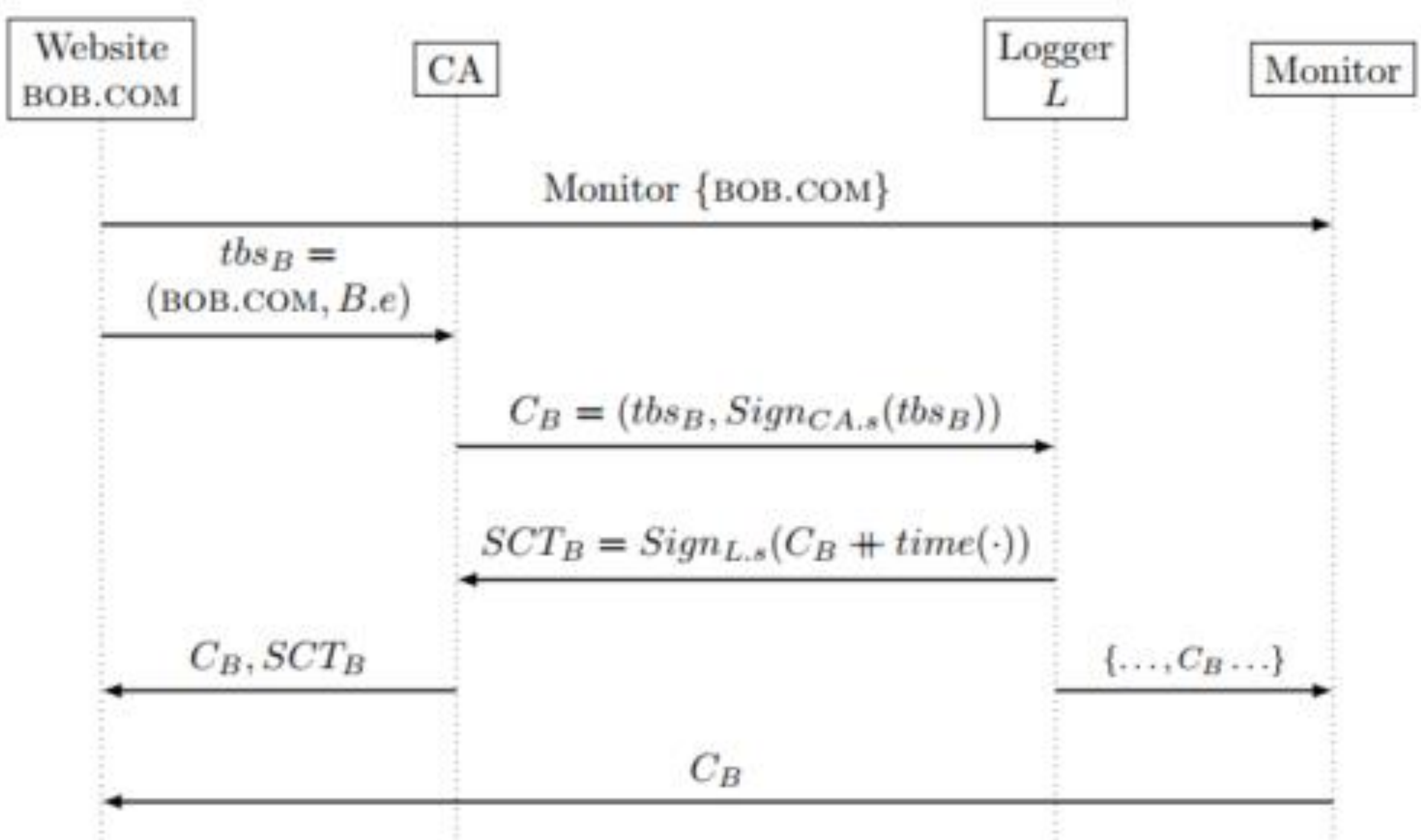
Monitor Detects Rogue Certificates

- Owner asks to monitor relevant domain names

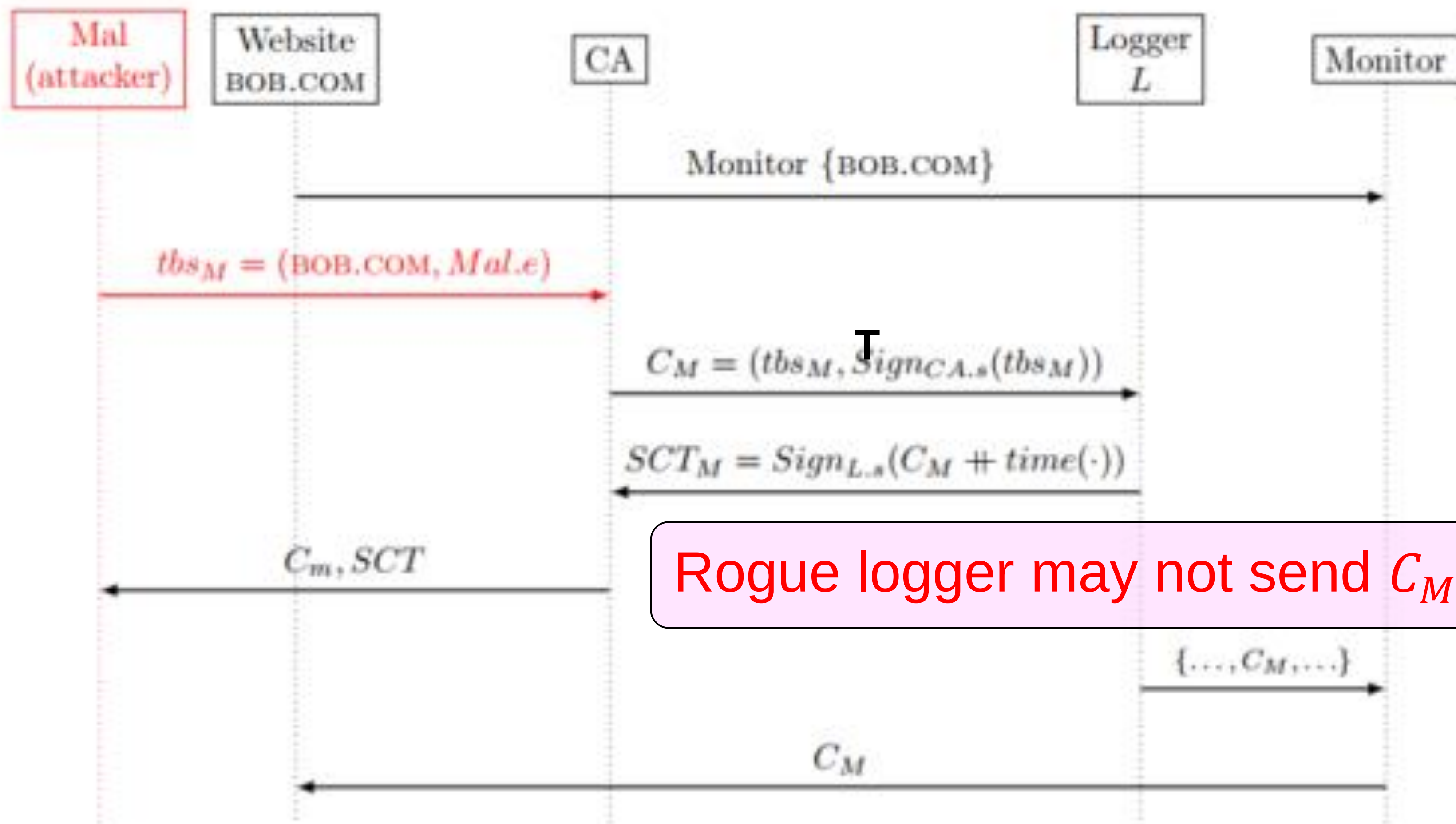


- Monitor asks for certs [Range, e.g., all new]
  - Usually periodically; assume daily (typical)
- Monitor sends to owner new certs for same domain name
  - Or suspect as misleading: combo, homographic, similar,…

Monitoring in Honest-Logger CT



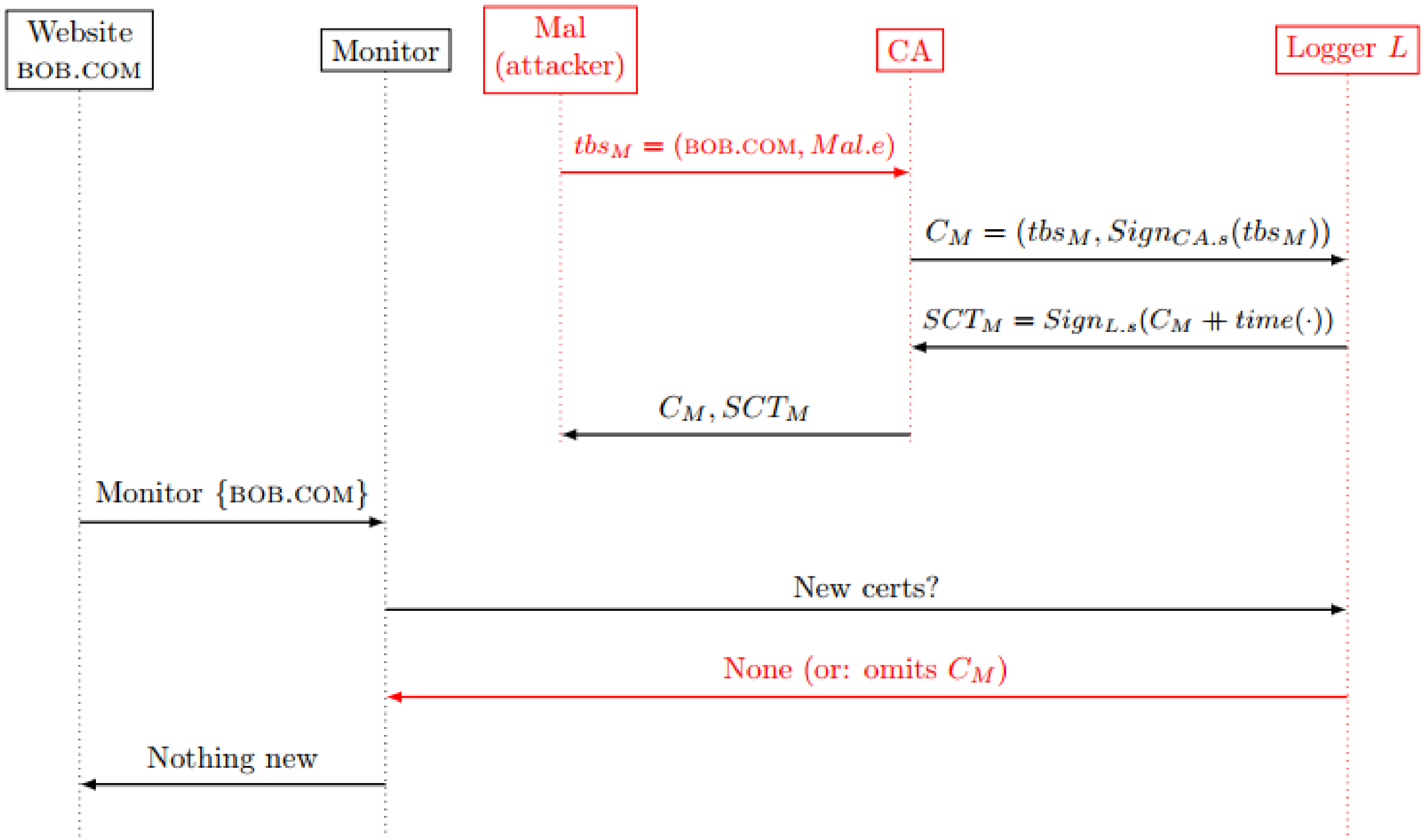
HL-CT: Detecting Rogue Certificate



Rogue logger may not send C_M

HL-CT: Omitted-Cert Attack by Rogue Logger

- Collusion of rogue CA and rogue Logger



Security against **Logger-CA Collusion**: 3 options

- **Option 1, redundancy**: SCTs signed by multiple loggers
  - How many loggers? Which loggers? Overhead ?
    - Google's Chrome: requires SCT from Google and one other SCT
      - Note: 'other' SCT is from logger chosen by (rogue?) CA…
      - 'In Google we Trust' ?
    - If relying party requires more redundancy, SCTs… good luck finding certificates! [Anti-trust?]
- Option 2, AnG-CT: Audit and Gossip CT
  - Heurist design to detect rogue loggers
  - Roughly follows RFC6962 and original CT publications
  - Complex, expose user privacy, …
- Option 3, NTTP-Secure CT (NS-CT):
  - Ensures `no trusted third party' by Proofs-of-Misbehavior (PoM)

Audit-and-Gossip (AnG) Certificate Transparency

- My interpretation of 'original' CT publications
  - Using Audit and Gossip to detect rogue loggers
  - No complete spec published so `extrapolating'
- Logger keeps certs in Merkle tree
  - Signed, timestamped digest: **Signed Tree Head (STH)**
  - Uses digest, PoI and **PoC (Proof-of-Consistency)** functions of the Merkle tree (or other Merkle digest scheme)

Merkle digest scheme: definition

Definition 4.15 (Merkle digest scheme). *A* Merkle digest scheme $\mathcal{M}$ *is a tuple of three PPT functions* $(\mathcal{M}.\Delta, \mathcal{M}.PoI, \mathcal{M}.VerPoI)$*, where:*

$\mathcal{M}.\Delta$ *is the* Merkle tree digest *function, whose input is a sequence of messages* $B = \{m_i \in \{0,1\}^*\}_i$ *and whose output is an* n*-bit* digest*:* $\mathcal{M}.\Delta : (\{0,1\}^*)^* \to \{0,1\}^n$.

$\mathcal{M}.PoI$ *is the* Proof-of-Inclusion *function, whose input is a sequence of messages* $B = \{m_i \in \{0,1\}^*\}_i$, an integer $i \in [1, |B|]$ *(the index of one message in* B*), and whose output is a* Proof-of-Inclusion *(PoI):* $\mathcal{M}.PoI : (\{0,1\}^*)^* \times \mathbb{N} \to \{0,1\}^*$.

$\mathcal{M}.VerPoI$ *is the* Verify-Proof-of-Inclusion *predicate, whose inputs are digest* $d \in \{0,1\}^n$, *message* $m \in \{0,1\}^*$, *index* $i \in \mathbb{N}$, *proof* $p \in \{0,1\}^*$, *and whose output is a bit (1 for 'true' or 0 for 'false'):* $\mathcal{M}.VerPoI : \{0,1\}^n \times \{0,1\}^* \times \mathbb{N} \times \{0,1\}^* \to \{0,1\}$.

Merkle Proof of Consistency (PoC)

- A Merkle digest scheme **supports PoC** if it has two more functions:

$\mathcal{M}.PoC(B_C, B_N)$ *is the* Extend and Proof-of-Consistency *function* PoC*, whose input are two sequences,* B_C *and* B_N*, and whose output* $\gamma_{CN} = \mathcal{M}.PoC(B_C, B_N)$ *is a binary string which we call the* Proof-of-Consistency *from* $\Delta_C \equiv \mathcal{M}.\Delta(B_C)$ *to* $\Delta_{CN} \equiv \mathcal{M}.\Delta(B_{CN})$.

$\mathcal{M}.VerPoC(\Delta_C, \Delta_{CN}, l_C, l_N, p) \in \{\mathbf{True}, \mathbf{False}\}$ *is the* Verify-Proof-of-Consistency *predicate, whose inputs are the two digests* Δ_C, Δ_{CN}*, the numbers of entries (*l_C *and* l_N*), and a string (*PoC*)* p.

- New digest Δ_{CN} is 'consistent' with current Δ_C
- I.e., is digest of block with the same first l_C messages, plus some l_N new messages

Merkle: Proof of Consistency (PoC)

- A Merkle digest scheme supports PoC if it has PoC, VerPoC functions
- Such scheme ensures correct PoC if :

$$\mathcal{M}.VerPoC\left(\mathcal{M}.\Delta(B_C), \mathcal{M}.\Delta(B_C + B_N), l_C, l_N, \mathcal{M}.PoC(B_C, B_N)\right) = \text{TRUE}$$

where $l_C = |B_C|$, $l_N = |B_N|$

- And ensures secure PoC if

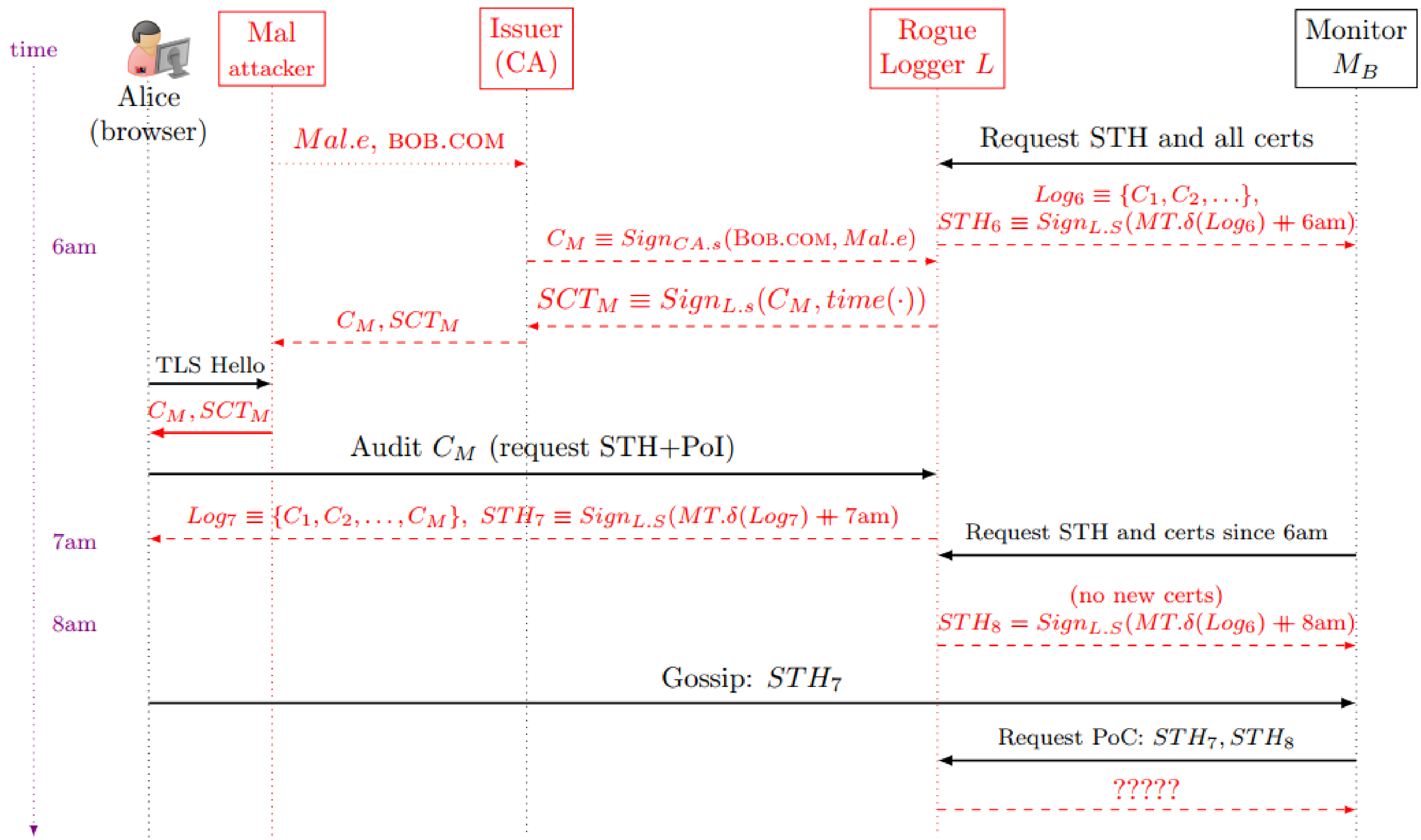
$$\varepsilon^{PoC}_{\mathcal{M},\mathcal{A}}(n) \equiv \Pr\left[\begin{array}{c} (B_C, B_A, l_C, l_A, p) \leftarrow \mathcal{A}(1^n) \ s.t. \\ \mathcal{M}.VerPoC(\mathcal{M}.\Delta(B_C), \mathcal{M}.\Delta(B_A), l_C, l_A, p) = \text{TRUE} \wedge \\ \wedge\ B_C \ \textit{is not a prefix of}\ B_A \end{array}\right]$$

is negligible, for every PPT adversary:

Audit-and-Gossip (AnG) Certificate Transparency

- My interpretation of 'original' CT publications
  - Using Audit and Gossip to detect rogue loggers
  - No complete spec published so `extrapolating'
- Logger keeps certs in Merkle tree
  - Signed, timestamped digest: **Signed Tree Head (STH)**
  - Uses digest, PoI and **PoC (Proof-of-Consistency)** functions
- Logger must respond to several **audit requests**:
  - Request for STH+PoI, for given certificate
  - Request for PoC, for given pair of STHs
  - Request for current STH
  - Request for certificates, logged between given start/end times
- **Gossip:** sharing of STHs among entities
  - To detect 'split world attack': different STHs to different entities

Audit-and-Gossip (AnG) Certificate Transparency

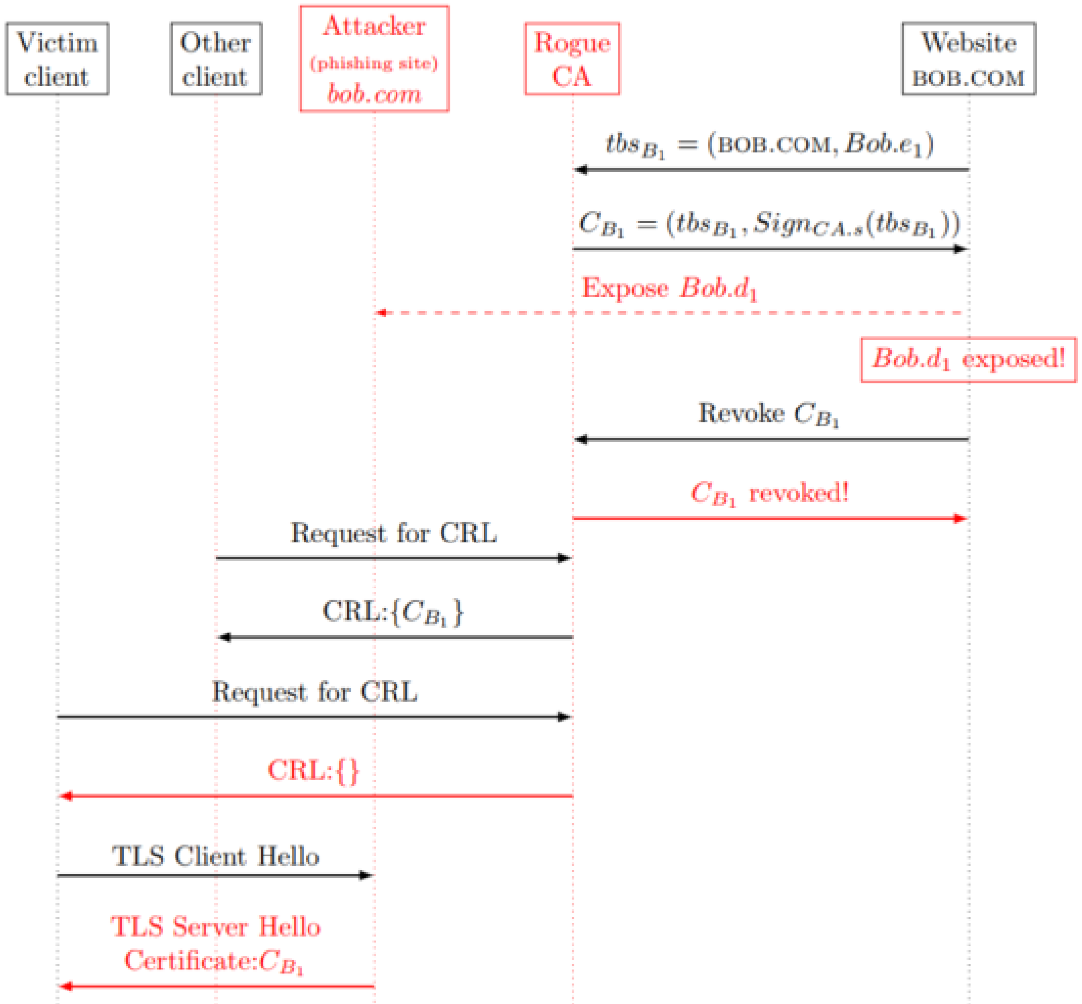


What is missing in AnG-CT ?

- May fail to provide Proof-of-Misbehavior (PoM)
  - Logger never sends the STH for a rogue SCT
  - Relying party receives no response… but no PoM!
  - Or, logger never responds to request for PoC for ‘rogue STH’…
  - **Goal: attacks are either ineffective or result in PoM**
    - **And: never a PoM against a honest party: no-false-PoM**
    - **Rigorously defined goal, for arbitrary protocols, using the Modular Security Specifications (MoSS) Framework – eprint 2020/1040**
- AnG’s Audit exposes sites visited by relying party to CA
  - Goal: preserve user’s privacy
- AnG-CT does not ensure revocation-status transparency
  - ➔ vulnerable to ‘zombie certificate attack’: mislead relying party into relying on a revoked certificate

The Zombie-Certificate Attack

- Rogue CA helps attacker by ‘un-revoking’ C_{B_1}
- Illustrated for CRL, similar for OCSP
- Against X.509, HL-CT, AnG-CT
- Foiled by NS-CT, since it ensures revocation-status transparency

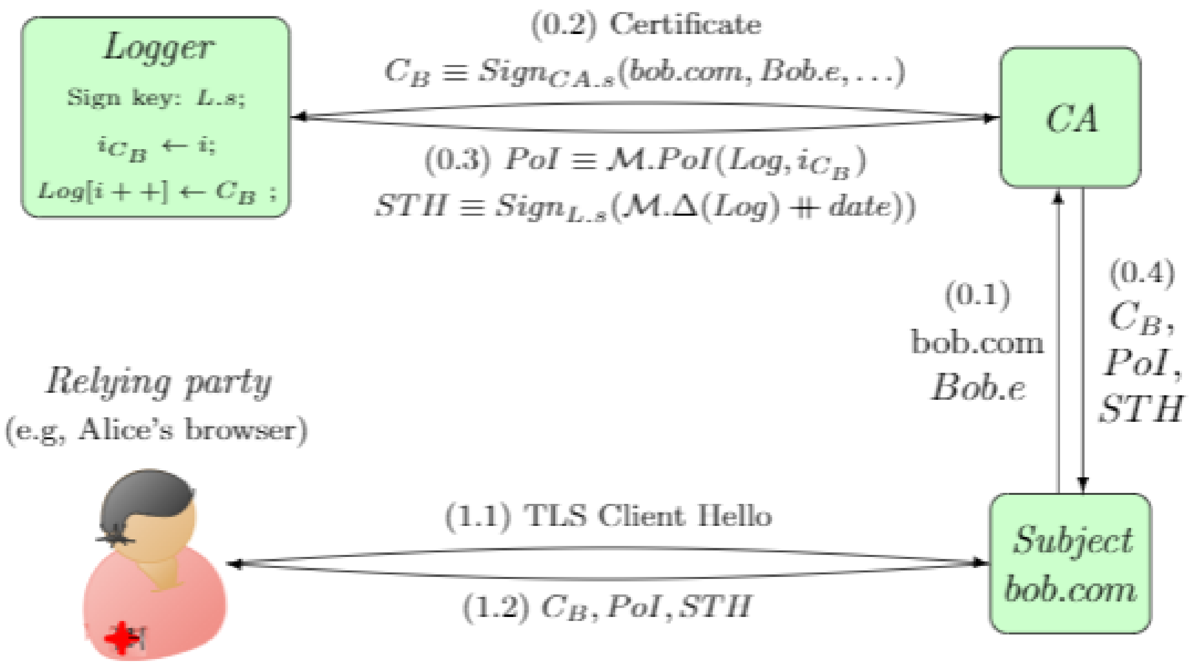


NS-CT (NTTP-Secure CT)

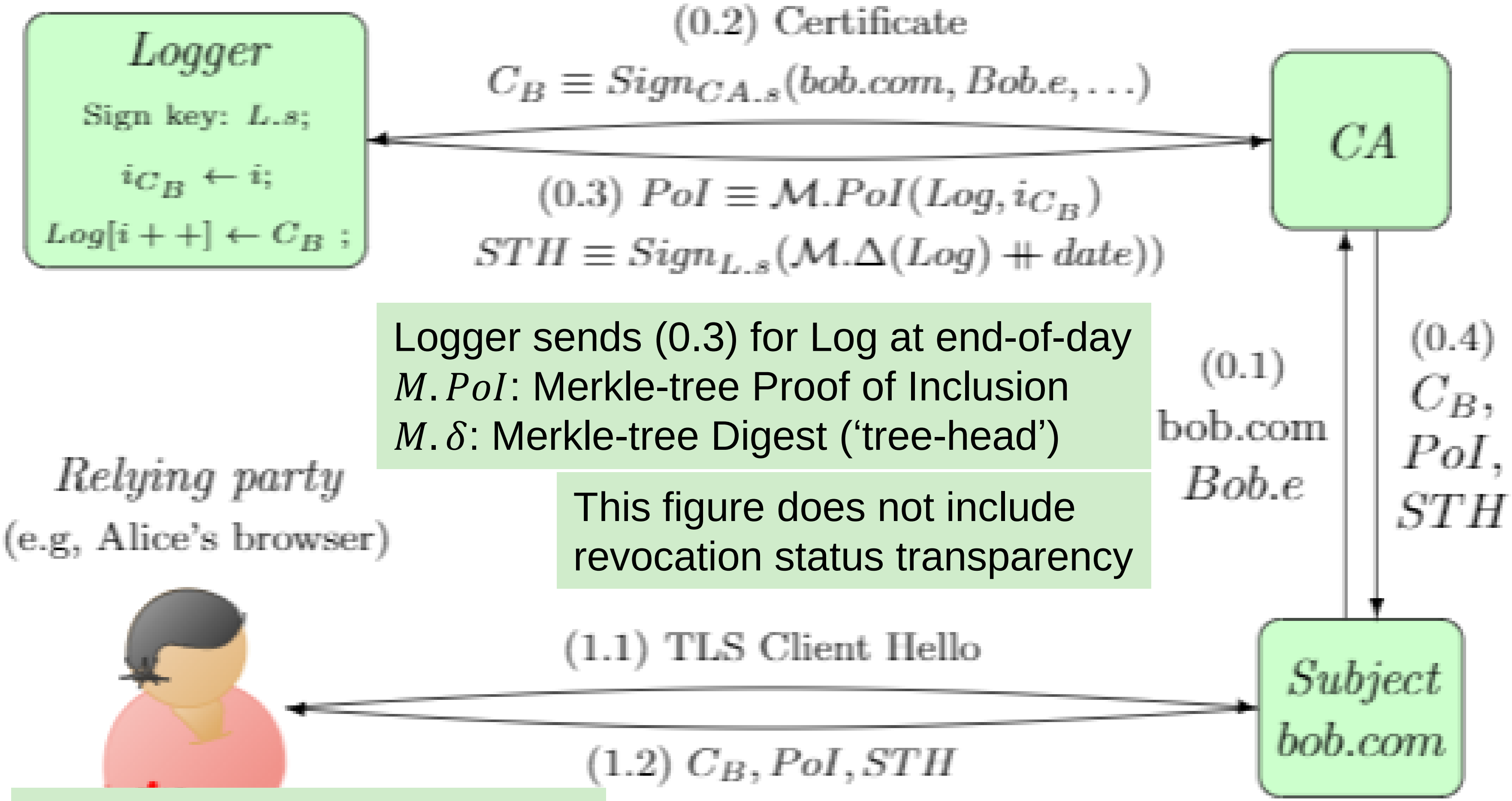
- **NTTP = No Trusted Third Party**
  - Secure against collusions of any set of parties (incl. loggers…)
  - Up to threshold t (maximal number of colluding parties)
- **Rogue certificate ➔ detection of rogue entity**
  - Monitors issue Proof-of-Misbehavior when rogue cert is audited
    - Certificate omitted from the log (or: invalid certificate in log)
    - Zombie-certificate – already revoked, and then 'resurrected'
  - **No false Proof-of-Misbehavior (PoM)**
    an honest entity is never considered corrupt
- Simplifications/assumptions:
  - Reliable communication between entities, synchronized clocks
    - We ignore delays and clock-skews, easy to handle these details
  - There are at least $2t + 1$ monitors (and at most t faulty).
  - All monitors observe all loggers (just for simplicity…)

NS-CT (NTTP-Secure CT) Issue Process

- Loggers issue **Signed Tree Head (STH)** every 24 hours
  - And provide it (within an hour) to all CAs, monitors
  - Response to CA includes STH and **Proof-of-Inclusion (PoI)**
  - CA, subject, relying party validate STH and PoI
  - Issue process almost unchanged – but takes up to 25 hours…



NTTP-Secure CT Issue Process: details

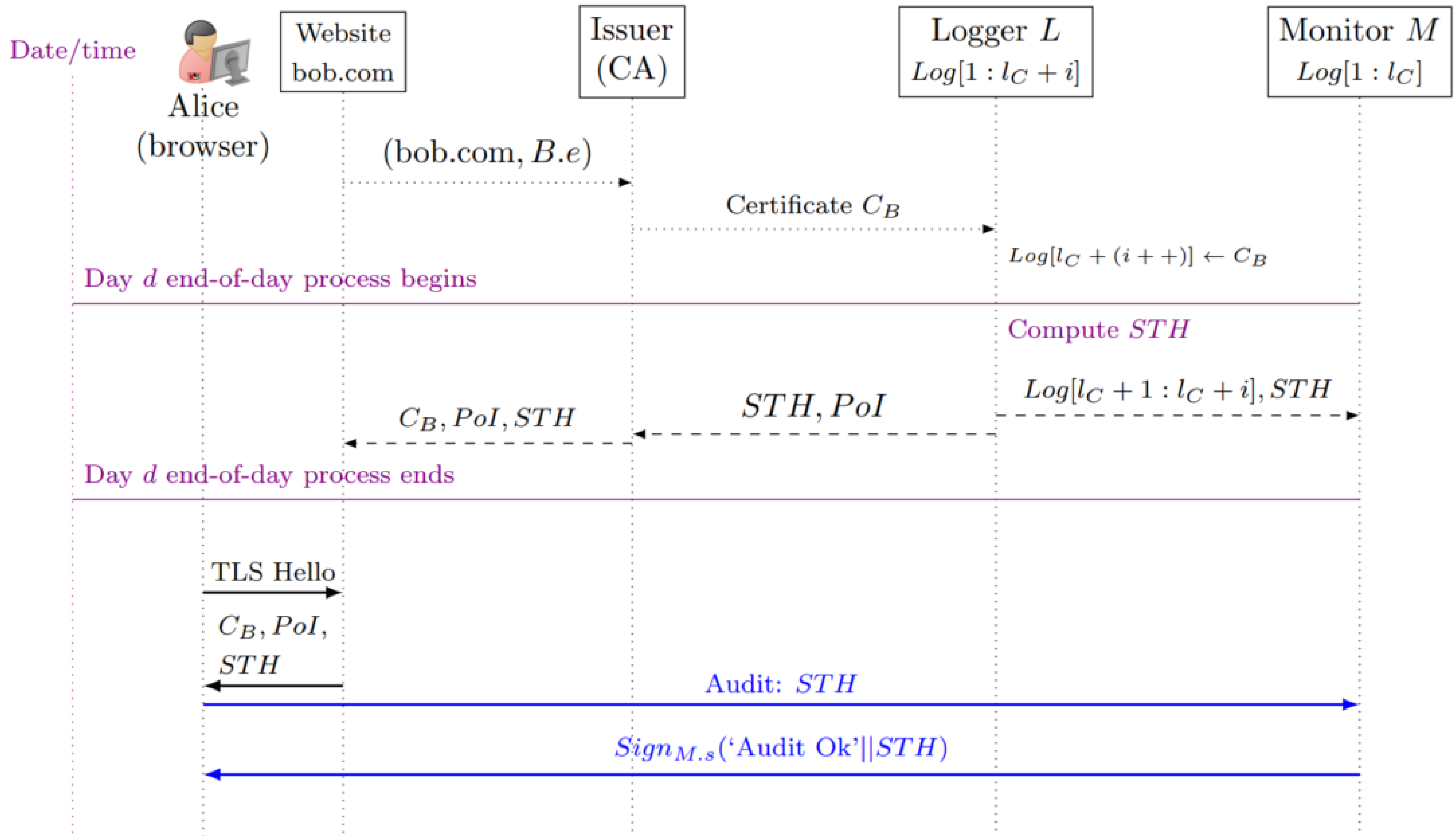


Logger sends (0.3) for Log at end-of-day
$M.PoI$: Merkle-tree Proof of Inclusion
$M.\delta$: Merkle-tree Digest ('tree-head')

This figure does not include revocation status transparency

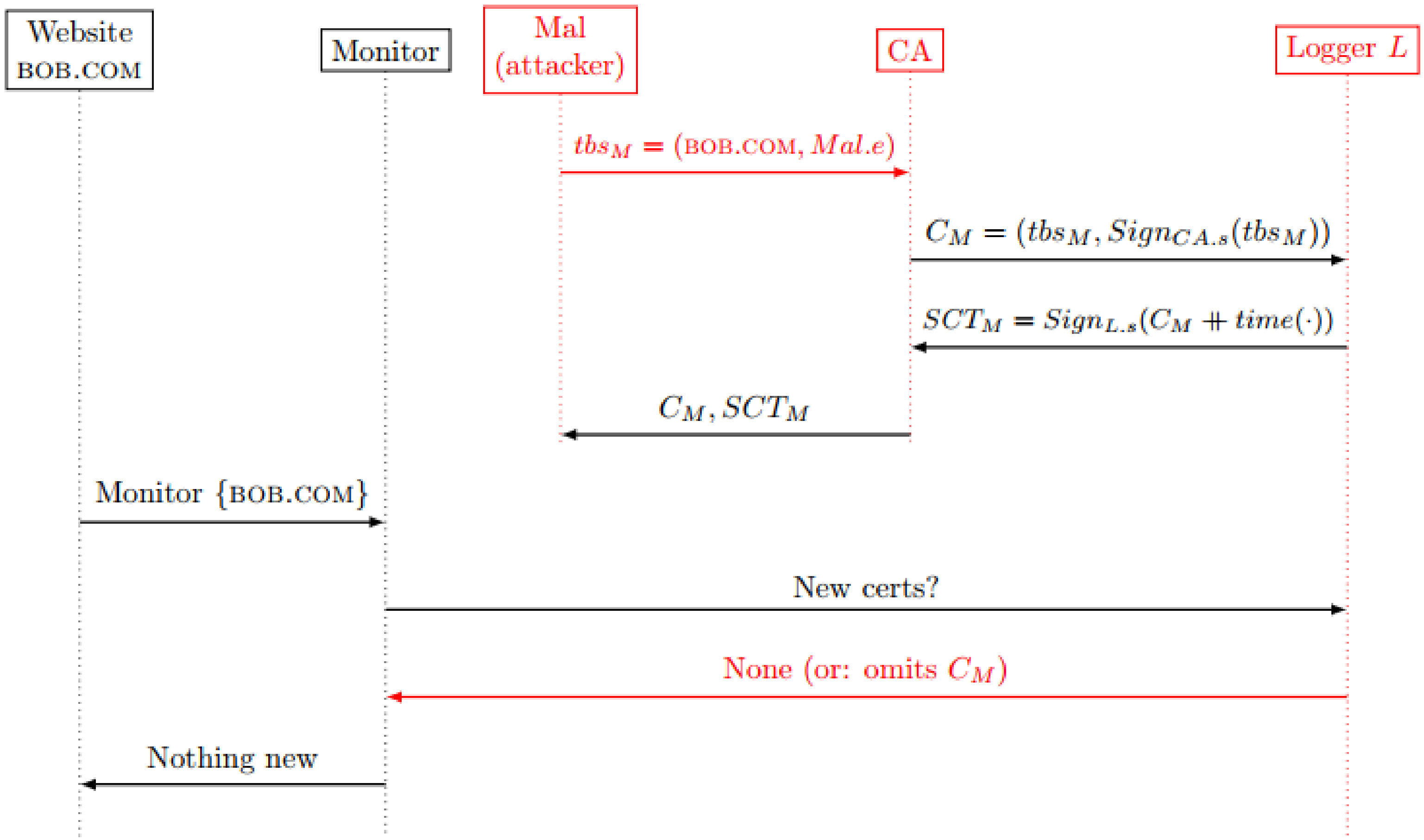
Alice validates signature on C_B and STH, date, and PoI

NS-CT : No-Faults Scenario

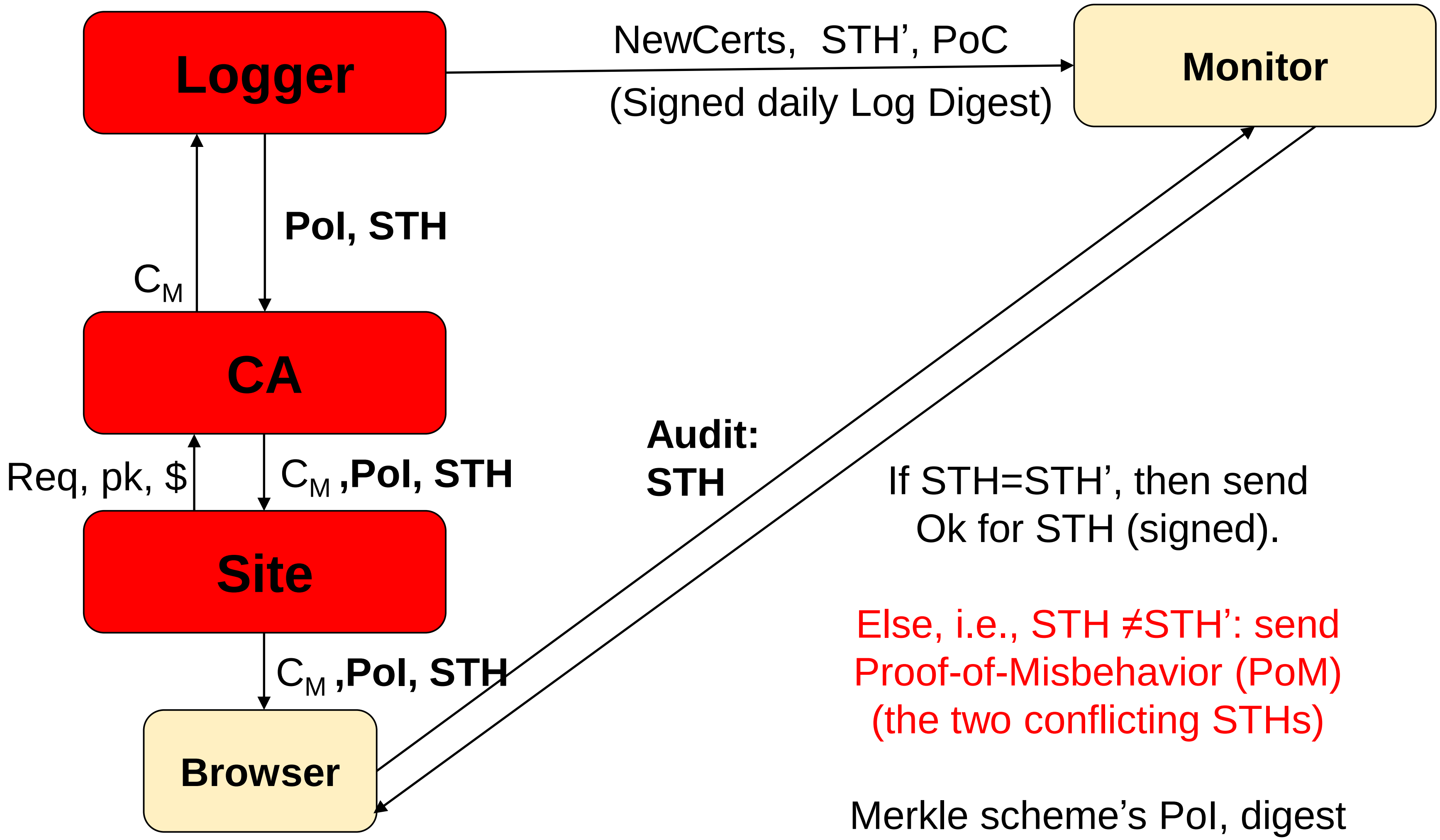


Recall Omitted-Cert Attack on HL-CT

- Collusion of rogue CA and rogue Logger



NS-CT: **Audit** detects omitted cert

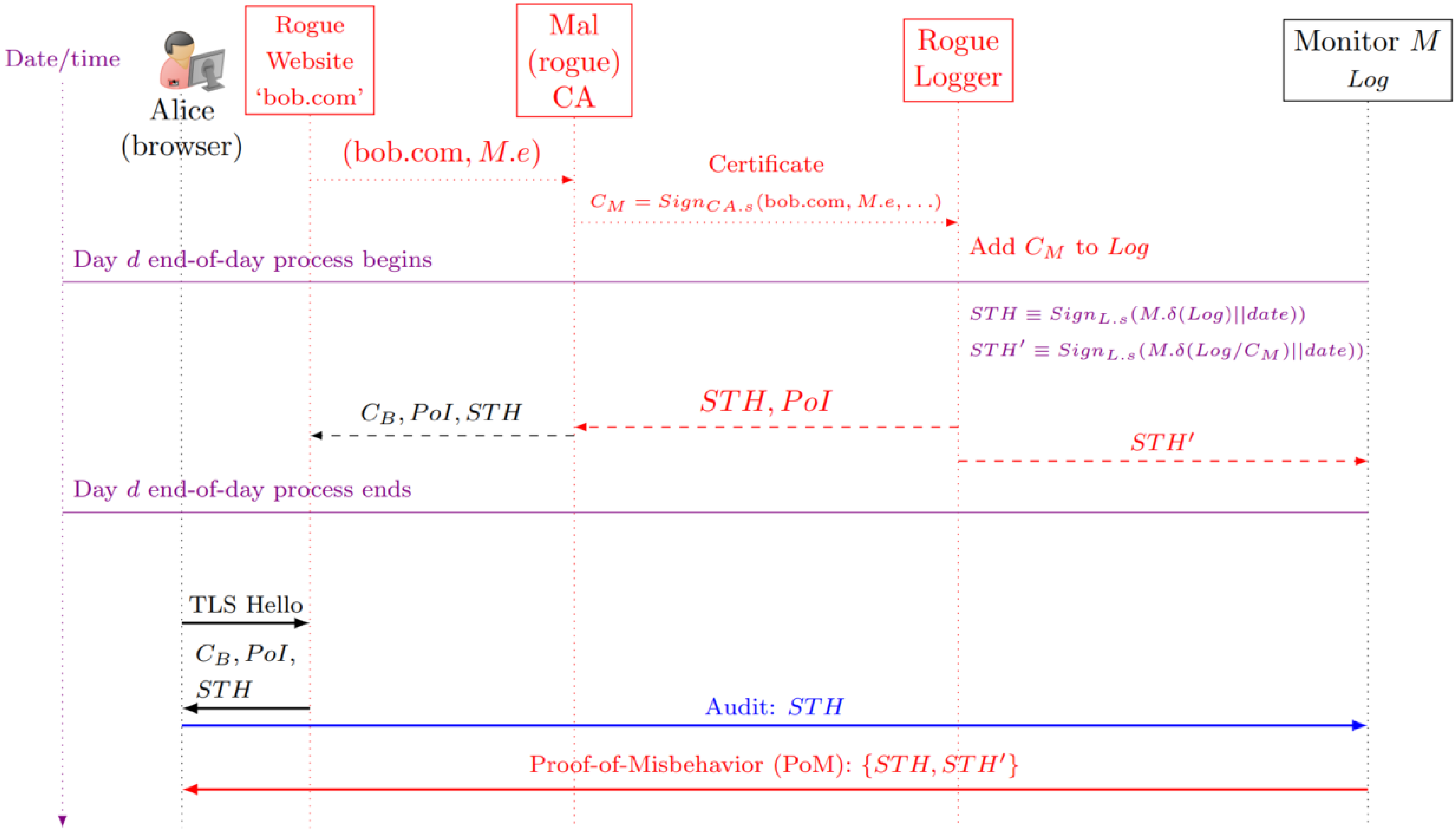


If STH=STH', then send Ok for STH (signed).

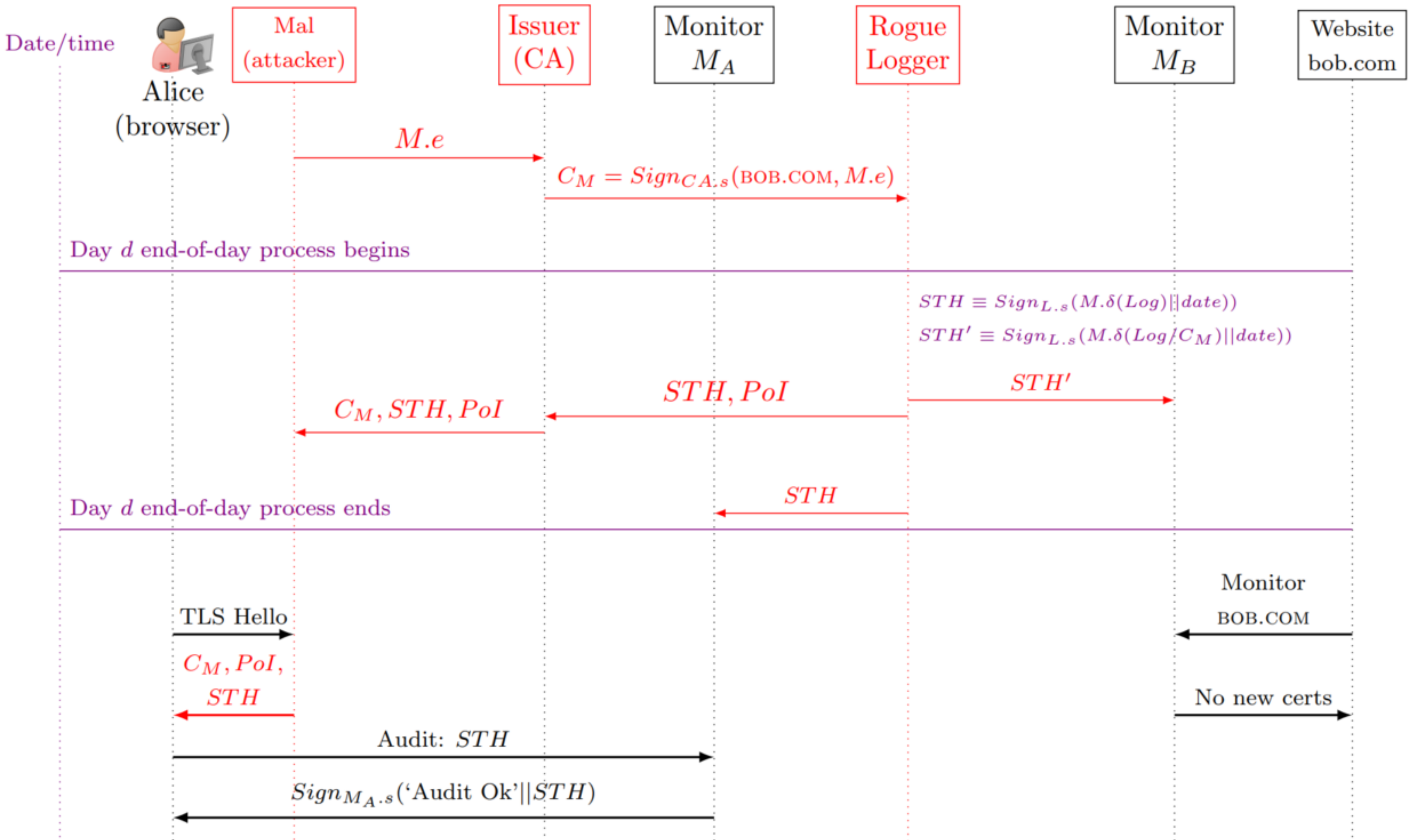
Else, i.e., STH ≠STH': send Proof-of-Misbehavior (PoM) (the two conflicting STHs)

Merkle scheme's PoI, digest properties ensure detection

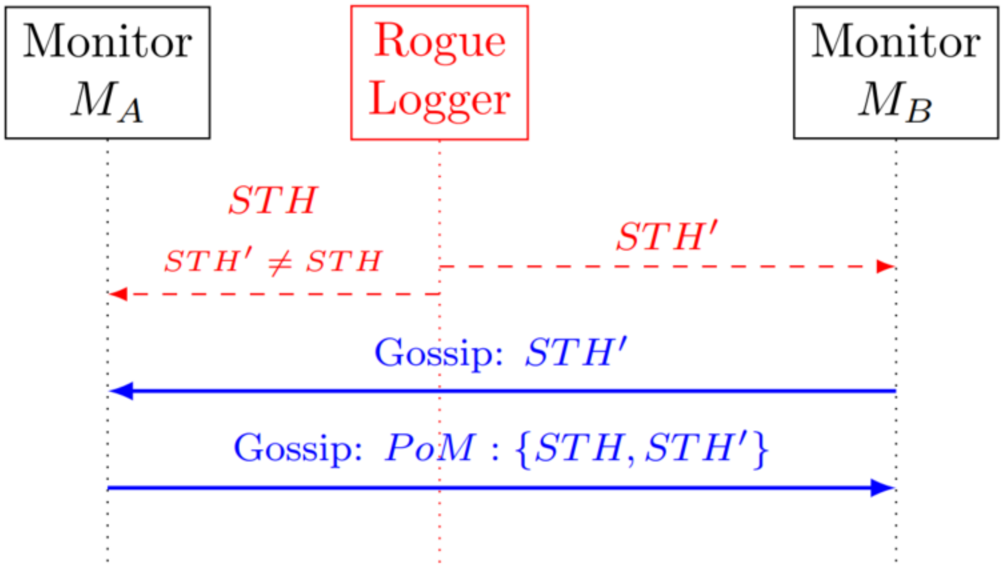
Proof-of-Misbehaving Logger: Omitted Cert.



NS-CT w/o Gossip: Split-World Attack



Inter-Monitor Gossip foils Split-World Attack



- Rogue logger may issue conflicting STHs:
  - STH_1: with rogue cert, sent to browser's monitor
  - STH_2: without rogue cert, sent to owner's monitor
- Gossip: detects, produce Proof-of-Misbehavior
- Detection occurs immediately (after receipt of STH) !

Summary: next generation of PKI

- Improved revocation
  - Stapled and/or pre-fetched; no online communication to CA
    - Preserve privacy
  - Efficient computations, communication
- Certificate and Certificate-Status transparency
  - Detect rogue certs for domain (same or misleading)
- NTTP (No Trusted-Third-Party) Security
  - Rogue certificate ➔ detection of rogue entity (PoM)
  - No false convictions (no false PoM)
- Not covered here:
  - Prevention/detection of equivocation
  - Definitions and proofs of security
    - Using the Modular Security Specs (MoSS) Framework