events in the system can be
- data access by user
- data access by third party
- data creation by user
**events** in the system can be
- data access by user
- data access by third party
- data creation by user

**transparency**: bad events are exposed
a transparency overlay

(architecture very much inspired by Certificate Transparency [LL’13])
a transparency overlay

(architecture very much inspired by Certificate Transparency [LL’13])
a transparency overlay

(architecture very much inspired by Certificate Transparency [LL’13])
a transparency overlay

(architecture very much inspired by Certificate Transparency [LL’13])
a transparency overlay

which systems?

system

GenEventSet

Log

CheckEntry

CheckEvidence

evidence

log server

log

Inspect

monitor

E
BE
snap

auditor

snap

Gossip

(architecture very much inspired by Certificate Transparency [LL’13])
a transparency overlay

(system)

GenEventSet

Log

CheckEvidence

which systems?

log server (log)

Inspect

monitor E BE snap

auditor snap

CheckEntry

Gossip
evidence

(architecture very much inspired by Certificate Transparency [LL’13])
design

system
design

system

log server log

log server log

log server log

log server log
design

system

log server log
design

system

GenEventSet

Log

log server

log
auditors (efficiently) determine if events are in the log
monitors (inefficiently) detect bad events in the log
auditors and monitors ensure consistent view of log
(can output evidence of inconsistencies)
system

GenEventSet

CheckEntry

Log

evidence

CheckEvidence

design

(add LS,Au,Mo)

security

monitor

E

BE

snap

auditor

snap

Inspector

Gossip

log server

log

which systems?
GenEventSet

Log

CheckEvidence

CheckEntry

Inspect

Gossip

Evidence

system

log_server

log

monitor

E, BE, snap

auditor

snap

which systems?

design

(add LS, Au, Mo)

security

construction
adversary wins if (1) **evidence fails** even though (2) monitor and auditor did have **inconsistent view**
non-frameability (related to [DGHS’16])

system

GenEventSet

Log

CheckEvidence

log server log

Inspect

monitor E BE snap

auditor snap

Gossip

evidence
non-frameability (related to [DGHS’16])

adversary wins if evidence passes
accountability

system

GenEventSet
Log
CheckEvidence

log server log

Inspect

monitor E BE snap

auditor snap

Gossip evidence
adversary wins if (1) it promised to include an event that (2) auditor and monitor believe to not be in the log, but (3) evidence fails
which systems?

system

GenEventSet

Log

CheckEvidence

Log server

Inspect

CheckEntry

monitor

E

BE

snap

auditor

snap

Gossip

evidence

monitor

E

BE

snap

auditor

snap

design

(add LS, Au, Mo)

security

(consistency)

(non-frameability)

(accountability)

construction
which systems?

system

GenEventSet

Log

CheckEvidence

design

(add LS,Au,Mo)

security

(consistency)

(non-frameability)

(accountability)

construction

Log server log

Inspect

monitor E BE snap

auditor snap

Gossip

evidence

which systems?
dynamic list commitment (dlc)

(aka tamper-evident log [CW’09])
(aka authenticated data structure [AGT’01,PSTY’13])
(aka rolling hash chain or Merkle tree [M’89])
dynamic list commitment (dlc)
dynamic list commitment (dlc)
dynamic list commitment (dlc)

basic
Com
CheckCom
Append

e₁ | e₂

system
dynamic list commitment (dlc)

**basic**
Com
CheckCom
Append

(generate succinct commitment)

```
\[ e_1 | e_2 \]
```

dotted line: system
dynamic list commitment (dlc)

basic
Com (generate succinct commitment)
CheckCom (check commitment)
Append

\[ e_1 e_2 \]
dynamic list commitment (dlc)

**basic**
- **Com**: (generate succinct commitment)
- **CheckCom**: (check commitment)
- **Append**: (add new events)

![Diagram showing dynamic list commitment](image)

```
  e1 e2 e3 e4
  --------------
  system
```
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

\[ e_1, e_2 \]

---

system
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend
  (can’t delete events)

```
[ e1 e2 e3 e4 ]
```

---

system
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

**specific event?**
- ProveIncl
- CheckIncl

*(can’t omit events)*

\[ e_1, e_2, e_3, e_4 \]

---

**system**
dynamic list commitment (dlc)

- **basic**
  - Com
  - CheckCom
  - Append

- **all events?**
  - ProveAppend
  - CheckAppend

- **specific event?**
  - ProveIncl
  - CheckIncl

```
e_1  e_2  e_3  e_4
```

---

system
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

**specific event?**
- ProveIncl
- CheckIncl

```
e_1 e_2 e_3 e_4
```

system
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

**specific event?**
- ProveIncl
- CheckIncl

This is ordered w.r.t. some notion of time.

```
e1 e2 e3 e4
```

system
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

**specific event?**
- ProveIncl
- CheckIncl

- inconsistent?
- DemoInconsistent
- CheckInInconsistent

"your commitment \( c \) does not represent the state of my list at time \( t \)"

this is ordered w.r.t. some notion of time
dynamic list commitment (dlc)

- **basic**
  - Com
  - CheckCom
  - Append

- **all events?**
  - ProveAppend
  - CheckAppend

- **specific event?**
  - ProveIncl
  - CheckIncl

```
e_1 e_2 e_3 e_4
```

- **inconsistent?**
  - DemoInconsistent
  - CheckInconsistent

- **non-inclusion?**
  - DemoNotIncl
  - CheckNotIncl

“your commitment c does not represent the state of my list at time t”

this is ordered w.r.t. some notion of time
construction

GenEventSet

Log

CheckEvidence

evidence

Log

Inspect

Gossip

CheckEntry

log server log

monitor E BE snap

auditor snap

system
construction

system

GenEventSet

log server log

Log

CheckEvidence

Inspect

CheckEntry

monitor E BE snap

auditor snap

Gossip

evidence
a (timed) signature, so no one can frame LS
a (timed) signature, so no one can frame LS
a (timed) signature, so LS is accountable
a (timed) signature, so LS is accountable
A (timed) signature, so LS is accountable.
log server

GenLog

Inspect

Gossip

log = E snap

snap = dlc t sig

Sys

Auditor

CheckEvidence

evidence

monitor

LS
log = E snap = dlc t sig

log server

GenEventSet

Inspect

Gossip

evidence

monitor

LS

Sys

event

Auditor

CheckEvidence

system Log

CheckEntry
log server

LS

Auditor

snap_A

event

Sys

update?

Auditor

t

dlc

log

E

snap

snap = dlc t sig

CheckEvidence

evidence
log server

**LS**

- **ProveAppend**
  - $\text{snap}_A$
  - $\text{snap}_{\text{LS,}\pi}$

**Auditor**

- **update?**

**Sys**

- **event**

**CheckEvidence**

- ** evidence**

- **update?**

Log:

- $\text{log} = \text{E} \quad \text{snap}$
- $\text{snap} = \text{dlc} \quad \text{t} \quad \text{sig}$
Log server → log

Inspect

log = E, snap
snap = dlc, t, sig

LS

find $E_\Delta$ (events since snap$_M$)

snap$_M$

Monitor

Gossip

GenEventSet
log server
log

Inspect

monitor
E
BE
snap

log = E
snap

snap = dlc

t

sig

system

GenEventSet

CheckEntry

GenEventSet

CheckEvidence

find $E_\Delta$ (events since $\text{snap}_M$)

$\text{snap}_{LS,E_\Delta}$

$\text{snap}_M$

Monitor

Gossip
Log server \( \log \)

Inspect

Monitor

\[ \text{Monitor} \]

\[ \text{Append}(E_{\Delta}, \text{dlc}_M) = \text{dlc}_L \text{S} \]?

use checks to update BE

\[ \text{find } E_{\Delta} \text{ (events since snap}_M \text{)} \]

\[ \text{snap}_M, E_{\Delta} \]

\[ \text{snap}_L, E_{\Delta} \]

log server \( \log \)

monitor \( E, \text{BE}, \text{snap} \)

log = \( E, \text{snap} \)

\( \text{snap} = \text{dlc, t, sig} \)
Monitor  Auditor

CheckEvidence

Gossip  evidence

log = E snap  snap = dlc t sig
Monitor \(\text{snap}_M, \text{snap}_A\) Auditor

Gossip \rightarrow \text{evidence}

\text{CheckEvidence}

\text{log} = E \quad \text{snap} = \text{dlc} t \quad \text{snap} = \text{sig}
Monitor $\text{snap}_M, \text{snap}_A$

DemolInconsistent($E, dlc_A, t_A$)

$\pi$

Monitor $\text{E, BE, snap}$

Auditor $\text{snap}$

CheckEvidence

Gossip

evidence

log = E snap = dlc t sig
Monitor | Auditor

\[ \text{snap}_M, \text{snap}_A \]

DemolInconsistent(\(E, \text{dlc}_A, t_A\))

\[ \pi \]

\(b \leftarrow \text{CheckInconsistent}(\text{dlc}_A, t_A, \text{dlc}_M, \pi)\)

if \(b\) return (\(\text{snap}_A, \text{snap}_M, \pi\))

\[ \log = E \quad \text{snap} \quad \text{snap} = \text{dlc} \quad \text{t} \quad \text{sig} \]
Monitor

\[ \text{snap}_M, \text{snap}_A \]

DemolInconsistent(\(E, \text{dlc}_A, t_A\))

\[ \pi \]

\[ \begin{align*} b &\leftarrow \text{CheckInconsistent}(\text{dlc}_A, t_A, \text{dlc}_M, \pi) \\ \text{if } b \text{ return (snap}_A, \text{snap}_M, \pi) \end{align*} \]

Auditor

CheckEvidence

checks that
(1) snapshots are signed by LS and
(2) \( \pi \) proves inconsistency

log = E snap

snap = dlc t sig

Monitor, Auditor, Gossip, evidence
security

ability to carry out DemoInconsistent, ProveAppend, and ProveIncl ⇒

consistency

unforgeability of DemoInconsistent, DemoNotIncl*, and signature scheme ⇒

non-frameability

ability to carry out DemoNotIncl* ⇒

accountability

*uses pledged version in which Auditor keeps track of failed events and gossips about them with Monitor to produce new type of evidence
goal: bad events are exposed
system receives promises to include events in the log

goal: bad events are exposed
system receives promises to include events in the log

+ auditors determine if these events are in the log

goal: bad events are exposed
system receives promises to include events in the log

+ auditors determine if these events are in the log

+ auditors and monitors ensure consistent view of log

goal: bad events are exposed
system receives promises to include events in the log

+ auditors determine if these events are in the log

+ auditors and monitors ensure consistent view of log
  ⇒ (by consistency + accountability)
  event is in monitor’s view of the log

goal: bad events are exposed
system receives promises to include events in the log

+ auditors determine if these events are in the log

+ auditors and monitors ensure consistent view of log
  ⇒ (by consistency + accountability)
  event is in monitor’s view of the log

+ monitors detect bad events in the log
  ⇒
  goal: bad events are exposed
system

GenEventSet

Log

CheckEntry

Inspect

log server log

CheckEvidence

design

(add LS,Au,Mo)

security

(consistency)

(non-frameability)

(accountability)

construction

(dlc+sig)

evidence

monitor E BE snap

auditor snap

Gossip

which systems?
which systems?

system

GenEventSet

Log

CheckEvidence

design

(add LS, Au, Mo)

security

(consistency)

(non-frameability)

(accountability)

monitor [E] [BE] snap

auditor [snap]

Inspect

Gossip

evidence

log server log

CheckEntry

(evidence)
Certificate Transparency

bad certificate issuance is exposed
⇒ clients are less likely to accept bad certificates

(icon by parkjisun from noun project)
Bitcoin

double spending is exposed
double spending is exposed ... provably!
Bitcoin

sender → miner → blockchain → receiver

log server → log

Log → CheckEvidence → evidence

send → receiver

double spending is exposed ... provably!
sender and receiver don’t need to store blockchain
double spending is exposed … provably!
sender and receiver don’t need to store blockchain
gives rise to hybrid system with no mining
open problems

system

GenEventSet

Log

CheckEntry

Inspect

log server log

monitor E BE snap

auditor snap

gossip

evidence

CheckEvidence

design

(add LS,Au,Mo)

security

(consistency)

(non-frameability)

(accountability)

construction

(dlc+sig)

(CT+Bitcoin)

which systems?
open problems

log server

GenEventSet

CheckEvidence

all parties needed?

monitor

Gossip

evidence

system

Log

CheckEntry

design

(add LS, Au, Mo)

security

(consistency)

(accountability)

construction

(dlc+sig)

CT+Bitcoin

which systems?

which systems?

(evidence)

monitor E BE snap

auditor snap

inspect

system
open problems

system

GenEventSet

Log

CheckEvidence

all parties needed?

monitor E BE snap

Gossip

evidence

auditor snap

CheckEnt privacy?

log server log

(CT+Bitcoin)

which systems?

log server snap

monitor E BE snap

auditor snap

(consistency)

(accountability)

(construction)

(dlc+sig)

design

(add LS,Au,Mo)

security

privacy?
open problems

system

GenEventSet

Log

CheckEvidence

all parties needed?

CheckEnt

privacy?

monitor

E

BE

snap

auditor

snap

Gossip

log server

log

CheckEvidence

(CT+Bitcoin)

which systems?

design

(add LS,Au,Mo)

security

(consistency)

(non-frameability)

(accountability)

construction

(dlc+sig)

open problems

all parties needed?

privacy?

better?
open problems

system

GenEventSet
Log

CheckEvidence

all parties needed?

CheckEnt privacy?

better?

log server log

monitor E BE snap

auditor snap

Gossip

(CT+Bitcoin)

others?

which systems?

design

(add LS,Au,Mo)

security

(consistency)

(non-frameability)

(accountability)

construction

(dlc+sig)
open problems

system

GenEventSet

Log

CheckEvidence

Log server

inspector

monitor E BE snap

auditor snap

Gossip

(CT+Bitcoin)

which systems?

others?

system

Thanks for listening!

Full version: eprint.iacr.org/2016/915

design

security

which systems?

all parties needed?

privacy?

better?

construction

(dlc+sig)

open problems

(consistency)

(non-frameability)

(accountability)

open problems

(consistency)

(non-frameability)

(accountability)
dynamic list commitment (dlc)
dynamic list commitment (dlc)

basic
Com
CheckCom
Append
dynamic list commitment (dlc)

basic
Com
CheckCom
Append

\[ e_1, e_2 \]
dynamic list commitment (dlc)

**basic**
Com
CheckCom
Append

\[
\text{Com}(e_1 e_2) = H(e_2 || H(e_1))
\]
dynamic list commitment (dlc)

**basic**

\[ \text{Com} \]

\[ \text{CheckCom} \]

\[ \text{Append} \]

\[
\text{Com} \left( \begin{array}{cc} e_1 & e_2 \end{array} \right) = H(e_2 || H(e_1))
\]

\[
\text{CheckCom} \left( c, \begin{array}{cc} e_1 & e_2 \end{array} \right) = (c = H(e_2 || H(e_1)))
\]
dynamic list commitment (dlc)

**basic**

*Com*

*CheckCom*

*Append*

\[
\text{CheckCom}(c, e_1 e_2) = (c = H(e_2 || H(e_1)))
\]

\[
\text{Append}(e_3 e_4, c_{12}) = H(e_4 || (H(e_3) || c_{12}))
\]

\[
\text{Com}(e_1 e_2) = H(e_2 || H(e_1))
\]
dynamic list commitment (dlc)

**basic**

Com

CheckCom

Append

\[
\text{Com} \left( \begin{array}{c}
e_1 \\
e_2
\end{array} \right) = H(e_2 || H(e_1))
\]

\[
\text{CheckCom} (c, \begin{array}{c}
e_1 \\
e_2
\end{array}) = (c = H(e_2 || H(e_1)))
\]

\[
\text{Append} \left( \begin{array}{c}
e_3 \\
e_4
\end{array}, c_{12} \right) = H(e_4 || (H(e_3) || c_{12}))
\]
dynamic list commitment (dlc)

basic
Com
CheckCom
Append

all events?
ProveAppend
CheckAppend

\[
\begin{align*}
\text{Com}(e_1, e_2) & = H(e_2 \| H(e_1)) \\
\text{CheckCom}(c, e_1, e_2) & = (c = H(e_2 \| H(e_1))) \\
\text{Append}(e_3, e_4, c_{12}) & = H(e_4 \| (H(e_3) \| c_{12}))
\end{align*}
\]
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

\[
\text{Com}(\begin{array}{c}e_1 \\ e_2\end{array}) = H(e_2 || H(e_1))
\]

\[
\text{CheckCom}(c, \begin{array}{c}e_1 \\ e_2\end{array}) = (c = H(e_2 || H(e_1)))
\]

\[
\text{Append}(\begin{array}{c}e_3 \\ e_4 \end{array}, c_{12}) = H(e_4 || (H(e_3) || c_{12}))
\]

\[
\text{ProveAppend}(c_{12}, c_{1234}, \begin{array}{c}e_1 \\ e_2 \\ e_3 \\ e_4\end{array}) = \begin{array}{c}e_3 \\ e_4\end{array}
\]
dynamic list commitment (dlc)

**Basic**
- Com
- CheckCom
- Append

**All Events?**
- ProveAppend
- CheckAppend

\[
\begin{align*}
\text{Com}(e_1, e_2) &= H(e_2 || H(e_1)) \\
\text{CheckCom}(c, e_1, e_2) &= (c = H(e_2 || H(e_1))) \\
\text{Append}(e_3, e_4, c_{12}) &= H(e_4 || H(e_3) || c_{12}) \\
\text{ProveAppend}(c_{12}, c_{1234}, e_1, e_2, e_3, e_4) &= e_3, e_4 \\
\text{CheckAppend}(c_{12}, c_{1234}, e_3, e_4) &= (c_{1234} = \text{Append}(e_3, e_4, c_{12}))
\end{align*}
\]
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

**specific event?**
- ProveIncl
- CheckIncl

\[
\text{Com}(e_1, e_2) = H(e_2 || H(e_1))
\]

\[
\text{CheckCom}(c, e_1, e_2) = (c = H(e_2 || H(e_1)))
\]

\[
\text{Append}(e_3, e_4, c_{12}) = H(e_4 || (H(e_3) || c_{12}))
\]

\[
\text{ProveAppend}(c_{12}, c_{1234}, e_1, e_2, e_3, e_4) = e_3, e_4
\]

\[
\text{CheckAppend}(c_{12}, c_{1234}, e_3, e_4) = (c_{1234} = \text{Append}(e_3, e_4, c_{12}))
\]
## dynamic list commitment (dlc)

<table>
<thead>
<tr>
<th>basic</th>
<th>all events?</th>
<th>specific event?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Com</td>
<td>ProveAppend</td>
<td>ProveIncl</td>
</tr>
<tr>
<td>CheckCom</td>
<td>CheckAppend</td>
<td>CheckIncl</td>
</tr>
<tr>
<td>Append</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Com**:
  \[
  \text{Com}(e_1, e_2) = H(e_2 || H(e_1))
  \]

- **CheckCom**: (c, e_1, e_2)
  \[
  \text{CheckCom}(c, e_1, e_2) = (c = H(e_2 || H(e_1)))
  \]

- **Append**: (e_3, e_4, c_12)
  \[
  \text{Append}(e_3, e_4, c_{12}) = H(e_4 || (H(e_3) || c_{12}))
  \]

- **ProveAppend**: (c_{12}, c_{1234}, e_1, e_2, e_3, e_4)
  \[
  \text{ProveAppend}(c_{12}, c_{1234}, e_1, e_2, e_3, e_4) = e_3, e_4
  \]

- **CheckAppend**: (c_{12}, c_{1234}, e_3, e_4)
  \[
  \text{CheckAppend}(c_{12}, c_{1234}, e_3, e_4) = (c_{1234} = \text{Append}(e_3, e_4, c_{12}))
  \]

- **ProveIncl**: (c_{1234}, e_3, e_1, e_2, e_3, e_4)
  \[
  \text{ProveIncl}(c_{1234}, e_3, e_1, e_2, e_3, e_4) = (c_{12}, e_4)
  \]
dynamic list commitment (dlc)

**basic**
- Com
- CheckCom
- Append

**all events?**
- ProveAppend
- CheckAppend

**specific event?**
- ProveIncl
- CheckIncl

\[
Com(\begin{array}{c} e_1 \\ e_2 \end{array}) = H(e_2||H(e_1))
\]

\[
CheckCom(c,\begin{array}{c} e_1 \\ e_2 \end{array}) = (c = H(e_2||H(e_1)))
\]

\[
Append(\begin{array}{c} e_3 \\ e_4 \\ c_12 \end{array}) = H(e_4||(H(e_3)||c_12))
\]

\[
ProveAppend(c_{12},c_{1234},\begin{array}{c} e_1 \\ e_2 \\ e_3 \\ e_4 \end{array}) = \begin{array}{c} e_3 \\ e_4 \end{array}
\]

\[
CheckAppend(c_{12},c_{1234},\begin{array}{c} e_3 \\ e_4 \end{array}) = (c_{1234} = Append(\begin{array}{c} e_3 \\ e_4 \\ c_12 \end{array}))
\]

\[
ProveIncl(c_{1234},e_3,\begin{array}{c} e_1 \\ e_2 \\ e_3 \\ e_4 \end{array}) = (c_{12},e_4)
\]

\[
CheckIncl(c_{1234},e_3,(c_{12},\begin{array}{c} e_4 \end{array})) = CheckAppend(c_{12},c_{1234},\begin{array}{c} e_3 \\ e_4 \end{array})
\]