Karlsruhe Institute of Technology

Practical and Robust Secure Logging from Fault-Tolerant Sequential Aggregate Signatures

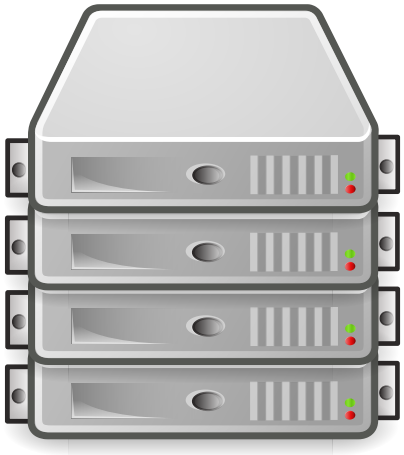
ProvSec 2017 | Gunnar Hartung, Björn Kaidel, Alexander Koch, Jessica Koch, Dominik Hartmann

DEPARTMENT OF INFORMATICS, INSTITUTE OF THEORETICAL INFORMATICS

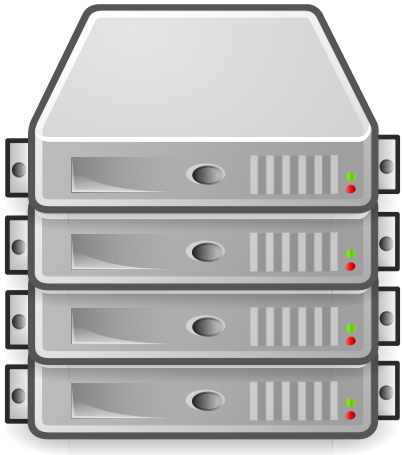

```
[   18.800138] init: cups main process (921) killed by HUP signal
[   18.800168] init: cups main process ended, respawning
[   19.277716] r8169 0000:03:00.0 eth1: link down
[   19.277736] r8169 0000:03:00.0 eth1: link down
[   19.277807] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[   19.278090] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[   19.454410] audit_printk_skb: 21 callbacks suppressed
[   19.454414] type=1400 audit(1452086079.075:19): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/sbin/dhclient" pid=1007 comm="apparmor_parser"
[   19.454421] type=1400 audit(1452086079.075:20): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=1007 comm="apparmor_parser"
[   19.454426] type=1400 audit(1452086079.075:21): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=1007 comm="apparmor_parser"
[   19.454742] type=1400 audit(1452086079.075:22): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/lib/x86_64-linux-gnu/lightdm-remote-session-freerdp/freerdp-session-wrapper" pi
d=1005 comm="apparmor_parser"
[   19.454751] type=1400 audit(1452086079.075:23): apparmor="STATUS" operation="profile_load" profile="unconfined" name="chromium" pid=1005 comm="apparmor_parser"
[   19.454776] type=1400 audit(1452086079.075:24): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=1007 comm="apparmor_parser"
[   19.454781] type=1400 audit(1452086079.075:25): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=1007 comm="apparmor_parser"
[   19.454866] type=1400 audit(1452086079.075:26): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/lib/lightdm/lightdm-guest-session" pid=1004 comm="apparmor_parser"
[   19.454870] type=1400 audit(1452086079.075:27): apparmor="STATUS" operation="profile_load" profile="unconfined" name="chromium" pid=1004 comm="apparmor_parser"
[   19.454977] type=1400 audit(1452086079.075:28): apparmor="STATUS" operation="profile_replace" profile="unconfined" name="chromium" pid=1005 comm="apparmor_parser"
[   20.269329] vboxdrv: module verification failed: signature and/or  required key missing - tainting kernel
[   20.272462] vboxdrv: Found 4 processor cores.
[   20.272850] vboxdrv: fAsync=0 offMin=0x4bd offMax=0x37a5
[   20.272936] vboxdrv: TSC mode is 'synchronous', kernel timer mode is 'normal'.
[   20.272937] vboxdrv: Successfully loaded version 4.3.34_Ubuntu (interface 0x001a000b).
[   20.288317] vboxpci: IOMMU not found (not registered)
[   20.968583] r8169 0000:03:00.0 eth1: link up
[   20.968594] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
[   22.513738] init: plymouth-upstart-bridge main process ended, respawning
[   22.526708] init: plymouth-upstart-bridge main process ended, respawning
[   48.266900] NFS: Registering the id_resolver key type
[   48.266913] Key type id_resolver registered
[   48.266915] Key type id_legacy registered
```

(Seal CC-0 by Tango Desktop Project)

Secure Logging – Motivation/First Try



Secure Logging – Motivation/First Try

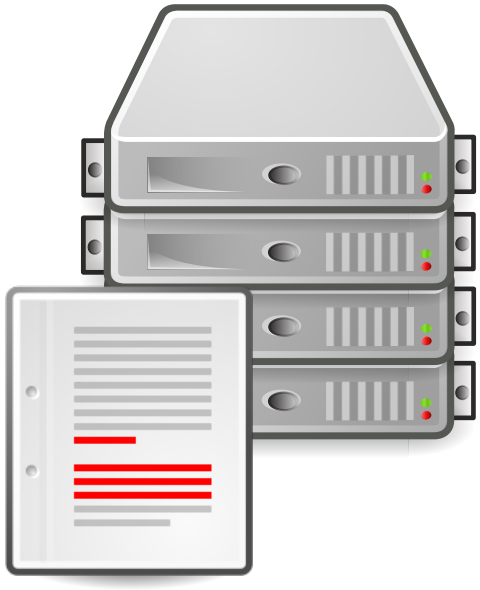


attacks stealthy

learns secrets

System under attack

Secure Logging – Motivation/First Try



attacks stealthy

learns secrets

We can use a log file to detect an attack later

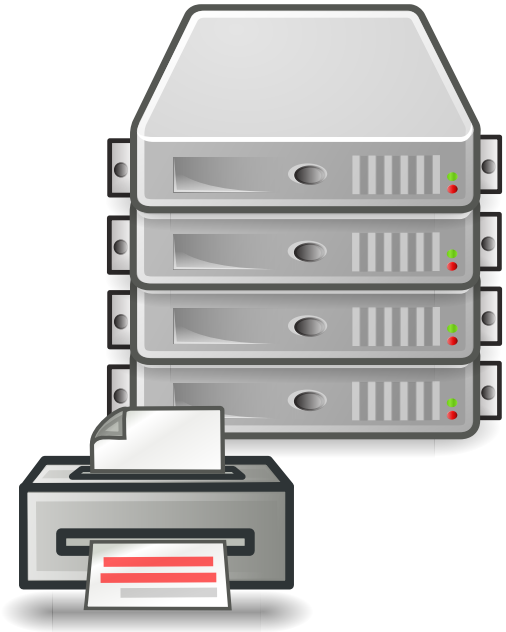
Secure Logging – Motivation/First Try

attacks stealthy

learns secrets

But the attacker may modify the log file to cover his traces!

Secure Logging – Motivation/First Try

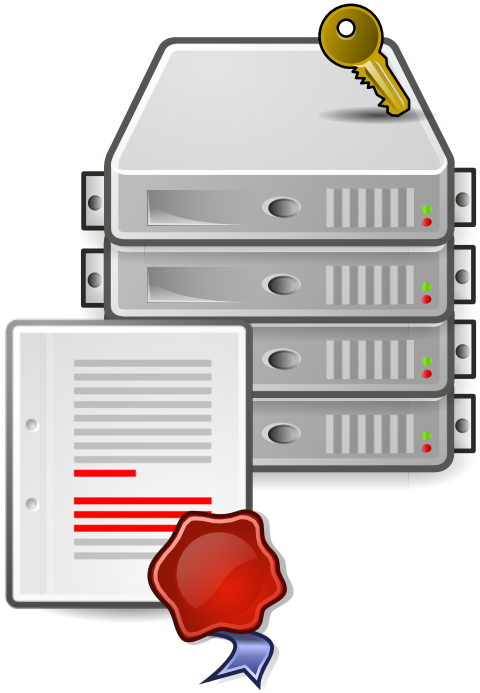


attacks stealthy

learns secrets

Hardware-solutions are costly, want to use crypto.

Secure Logging – Motivation/First Try

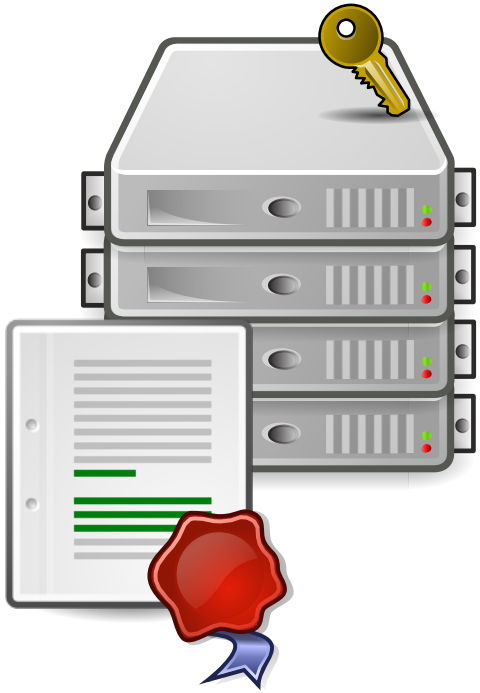


attacks stealthy

learns secrets

Lets use a signature to protect the log file.

Secure Logging – Motivation/First Try



attacks stealthy

learns secrets

But the attacker may retrieve the signing key and just re-sign!

Secure Logging – Motivation/First Try

“I am scared!”

Aim: We want a provably secure logging scheme that features

1. forward-security
2. truncation-resistance
3. robustness (while retaining space-efficiency)

Forward-Secure Model

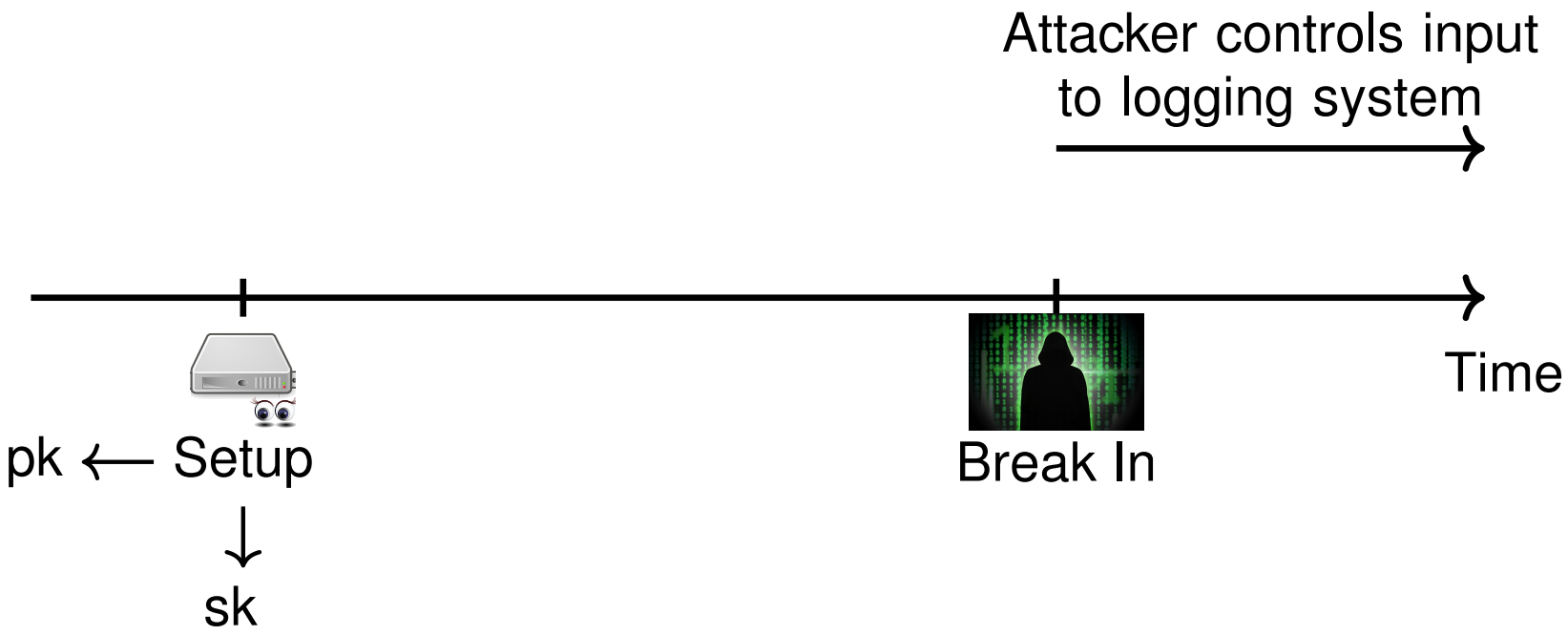
Idea: Sign log entries using forward-secure signatures [BM99]

Forward-Secure Model

Idea: Sign log entries using forward-secure signatures [BM99]

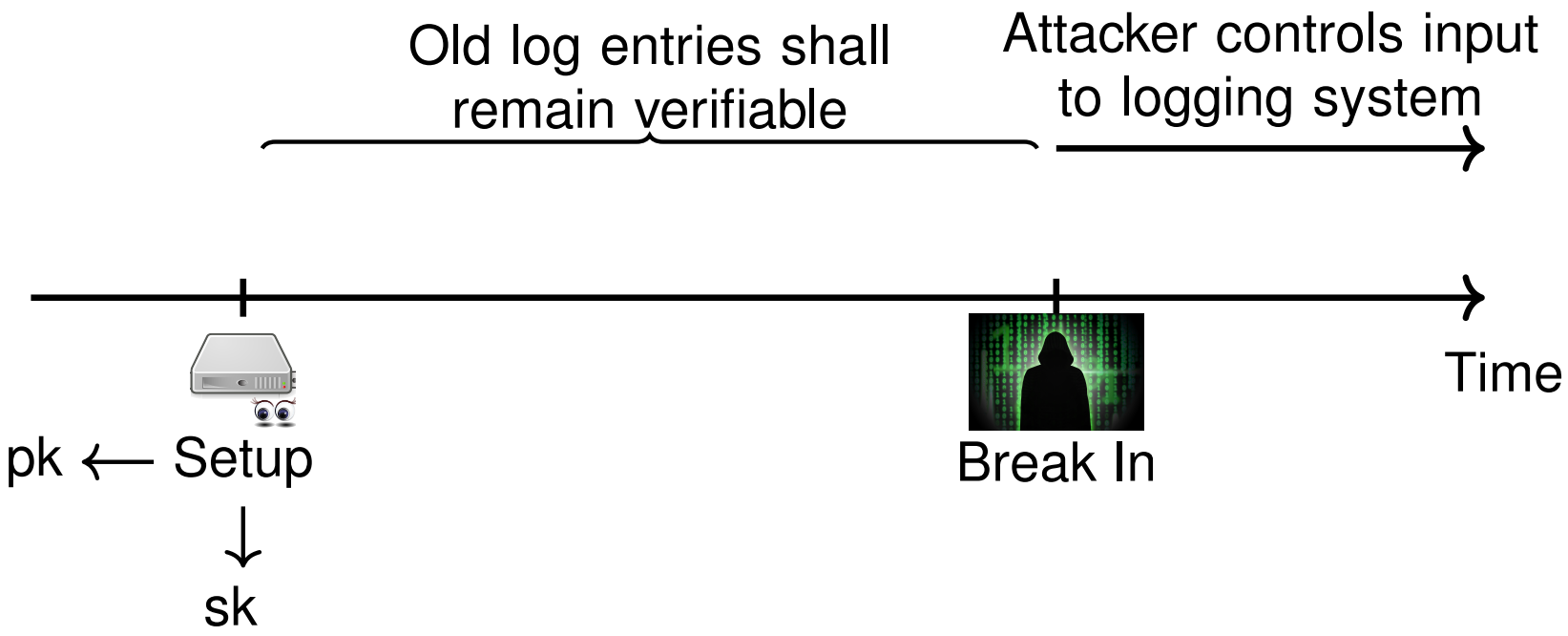
Forward-Secure Model

Idea: Sign log entries using forward-secure signatures [BM99]



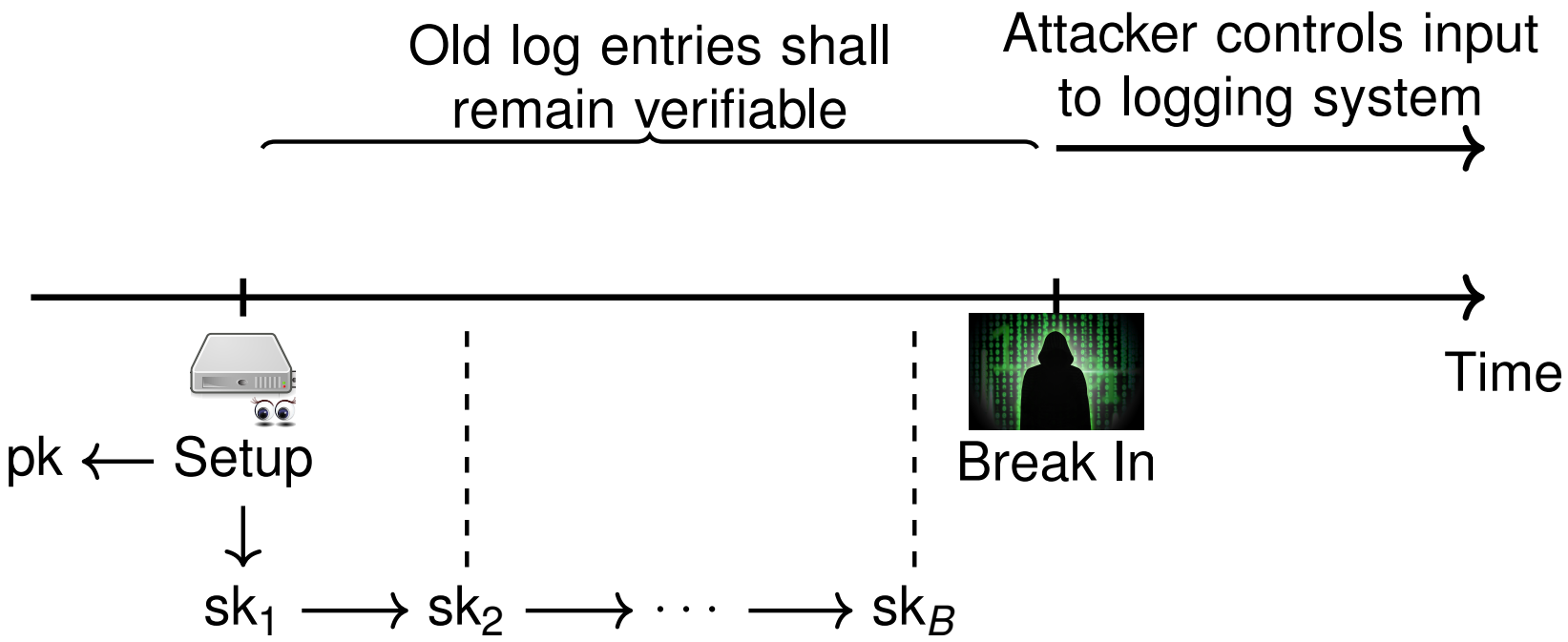
Forward-Secure Model

Idea: Sign log entries using forward-secure signatures [BM99]



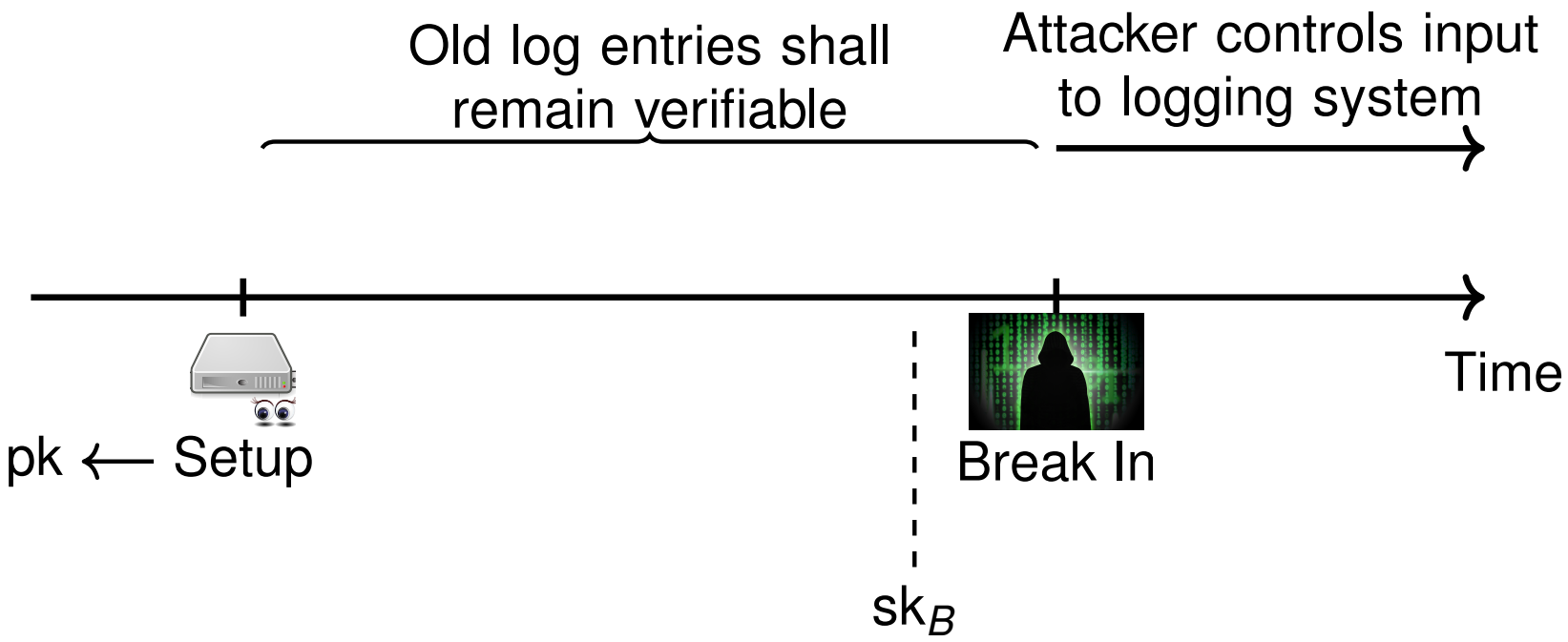
Forward-Secure Model

Idea: Sign log entries using forward-secure signatures [BM99]



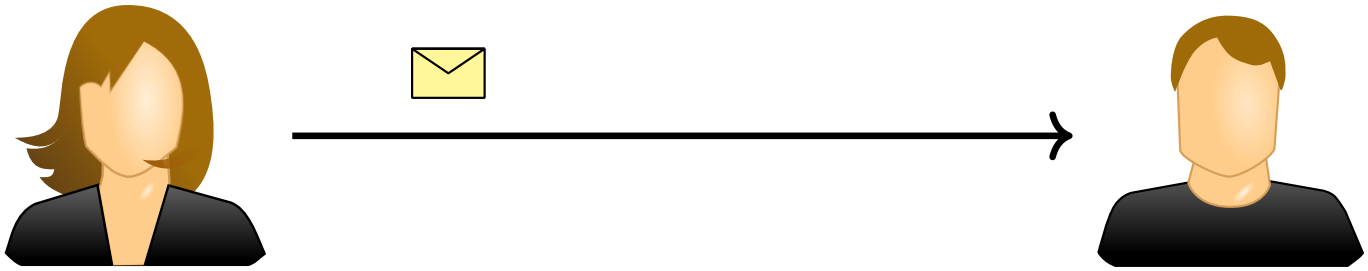
Forward-Secure Model

Idea: Sign log entries using forward-secure signatures [BM99]



Achieving truncation-resistance [MT08]

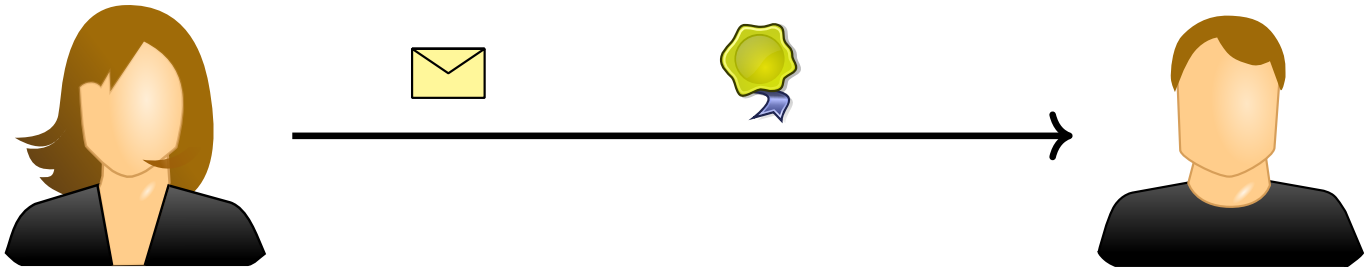
W/ [forward-secure sequential] aggregate signatures [Bon+03]



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Achieving truncation-resistance [MT08]

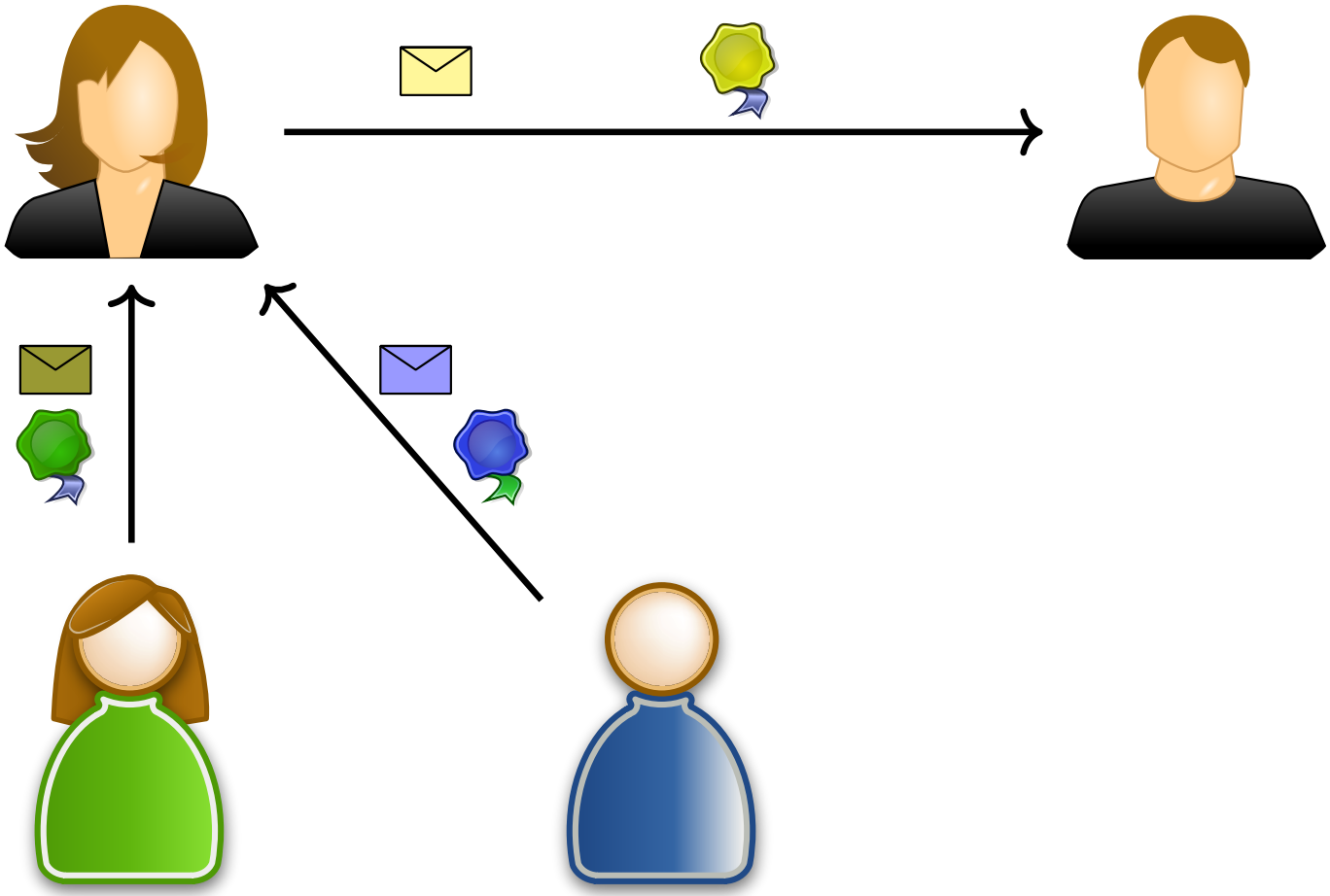
W/ [forward-secure sequential] aggregate signatures [Bon+03]



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

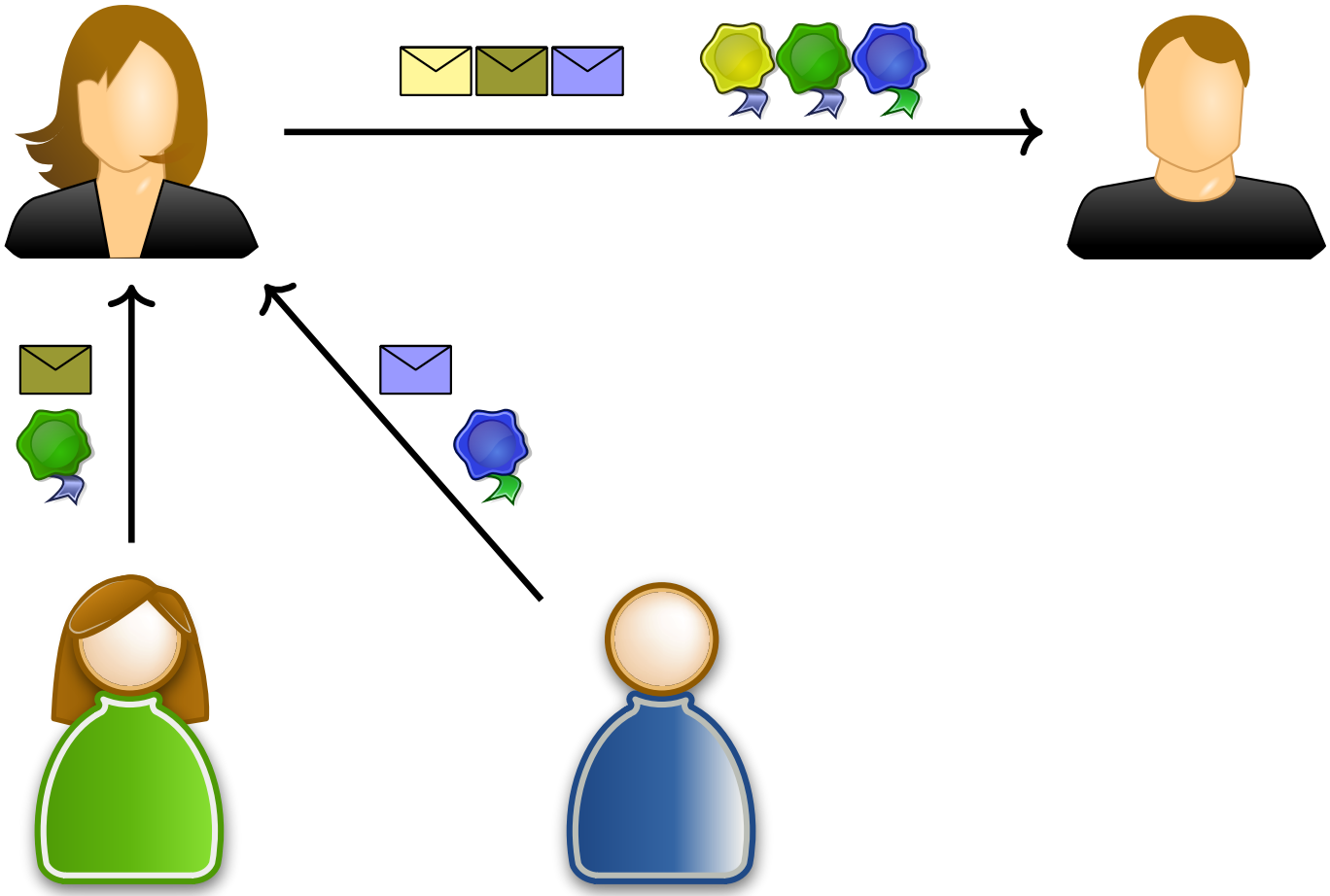
Achieving truncation-resistance [MT08]

W/ [forward-secure sequential] aggregate signatures [Bon+03]



Achieving truncation-resistance [MT08]

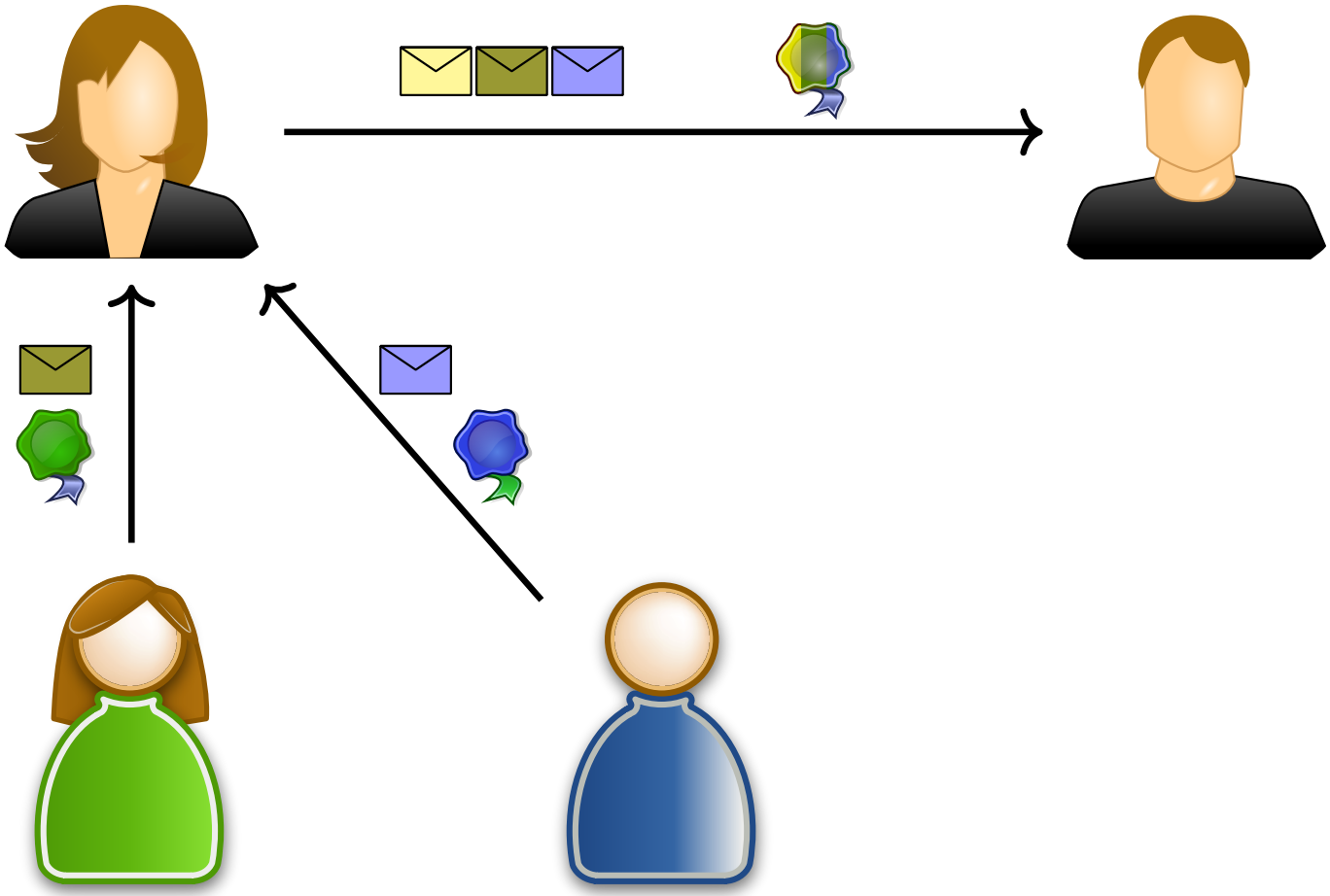
W/ [forward-secure sequential] aggregate signatures [Bon+03]



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Achieving truncation-resistance [MT08]

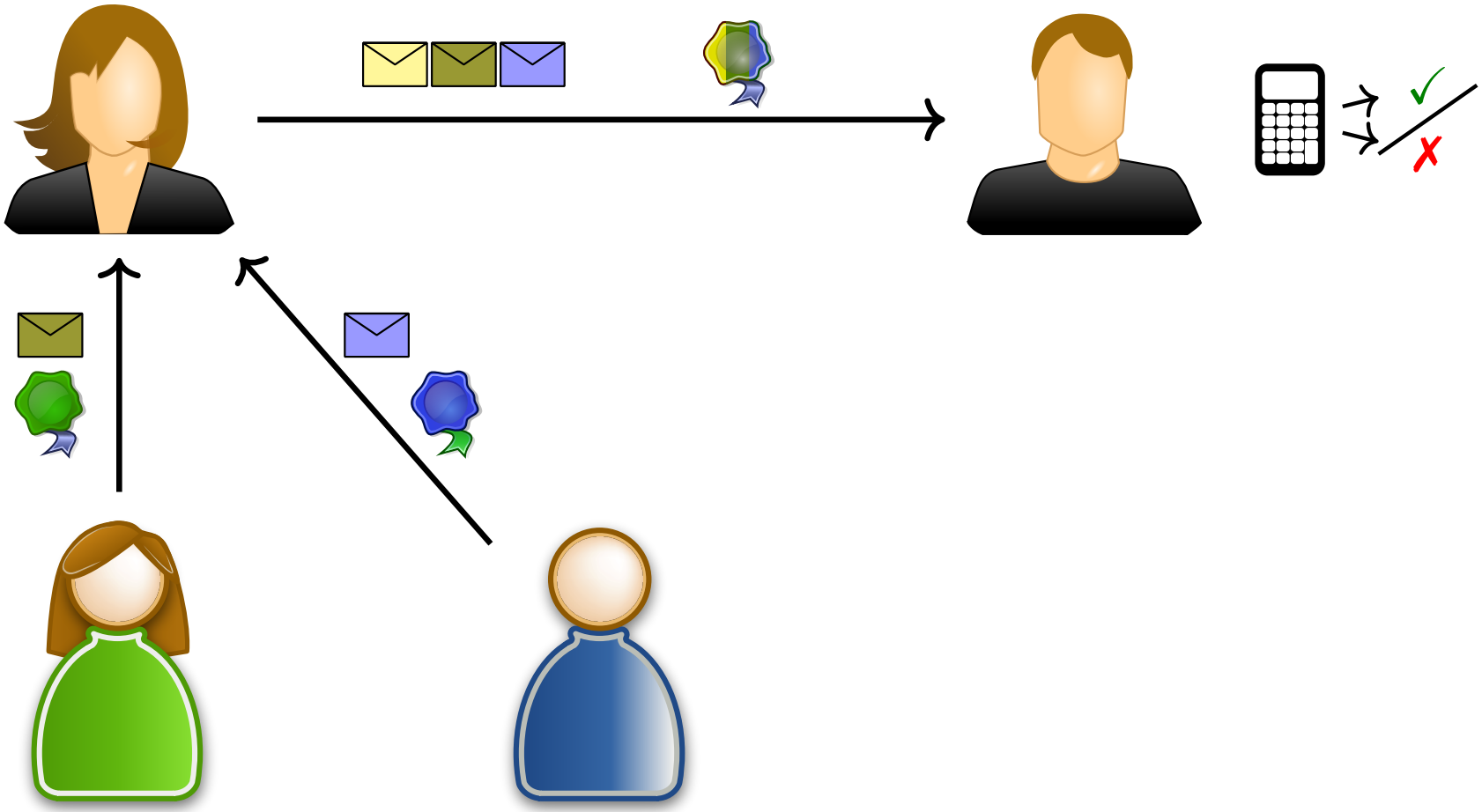
W/ [forward-secure sequential] aggregate signatures [Bon+03]



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Achieving truncation-resistance [MT08]

W/ [forward-secure sequential] aggregate signatures [Bon+03]



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Robustness for Aggregate Signatures

Karlsruhe Institute of Technology

What happens if verification fails?

Robustness for Aggregate Signatures

What happens if verification fails?

Bob doesn't know which/how many messages are invalid

$\Longrightarrow$ If we have an invalid aggregate signature for our logfile, everything is untrusted! This makes it very non-robust to log-modifications

Possible fix: Save all log entry signatures also.

Robustness for Aggregate Signatures

What happens if verification fails?
Bob doesn't know which/how many messages are invalid
$\Longrightarrow$ If we have an invalid aggregate signature for our logfile, everything is untrusted! This makes it very non-robust to log-modifications

Possible fix: Save all log entry signatures also.

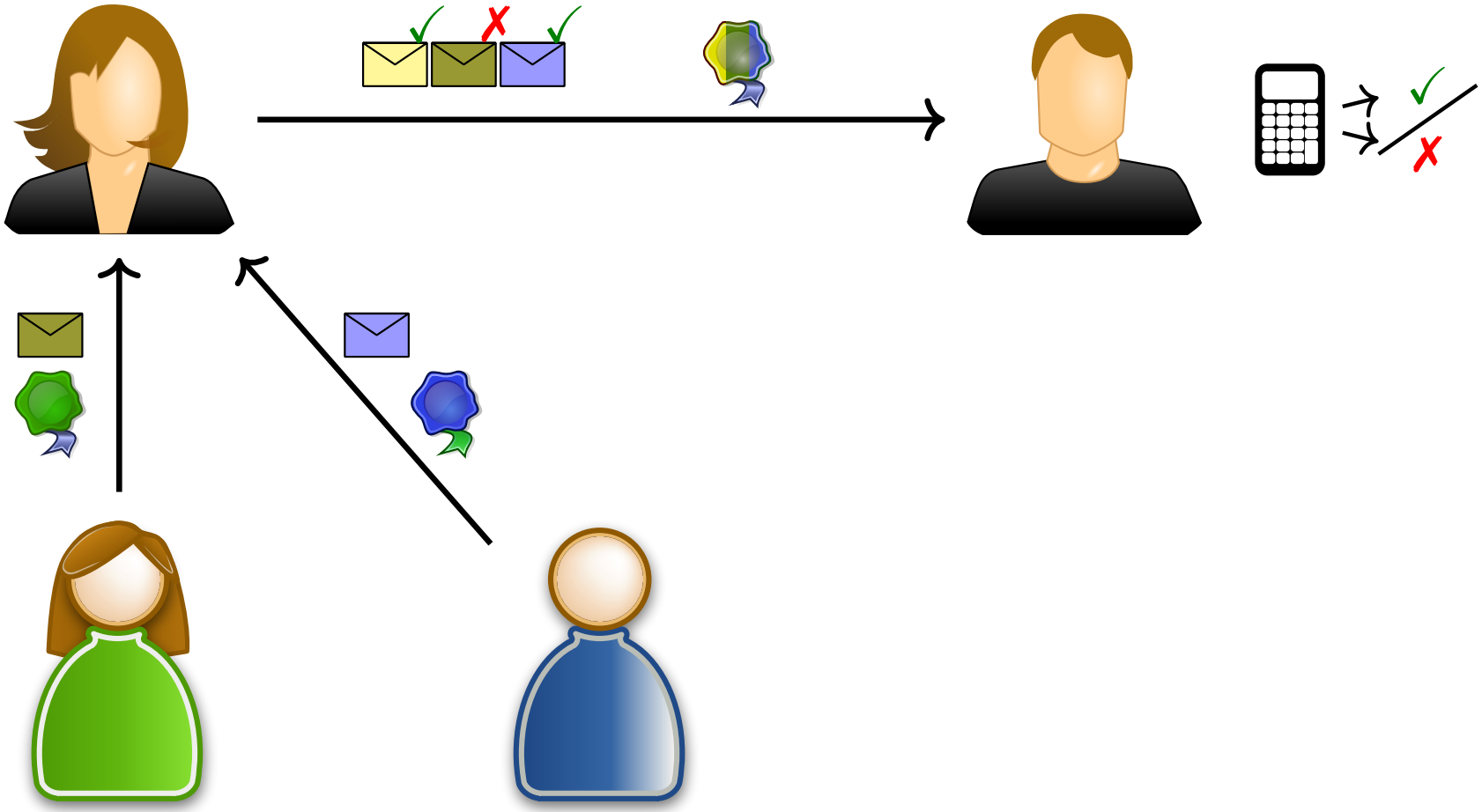
However, this is

- not space-efficient (uses n sigs, want: $\log(n)$)
- it might hinder truncation-security!
  For general aggregation: adversary just reaggregates all sigs excluding the entries to be truncated

We can do better :-)

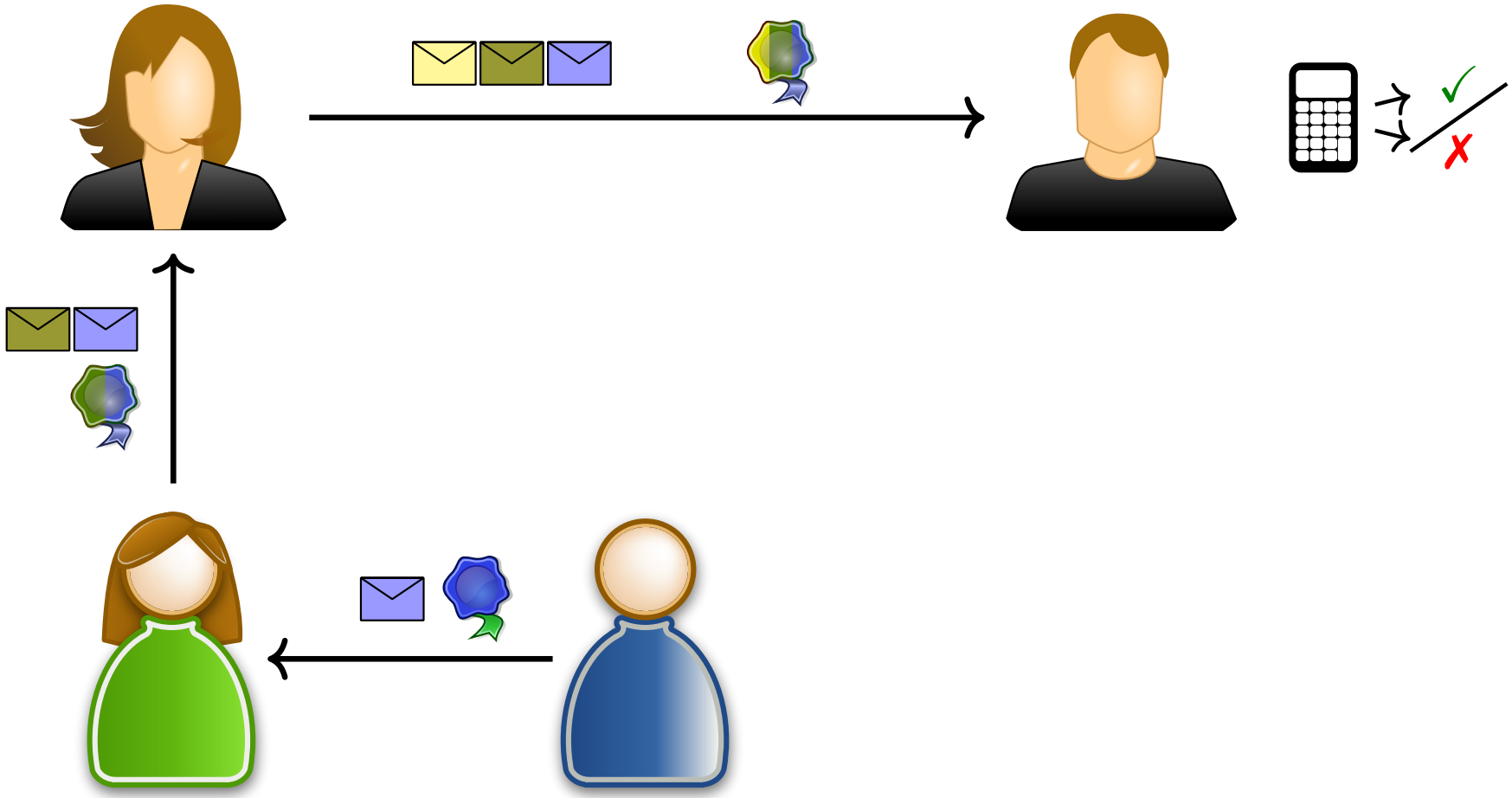
Adding Robustness to Aggregate Signatures

Hartung et al. [Har+16]



Relaxed: Sequential Aggregate Signatures

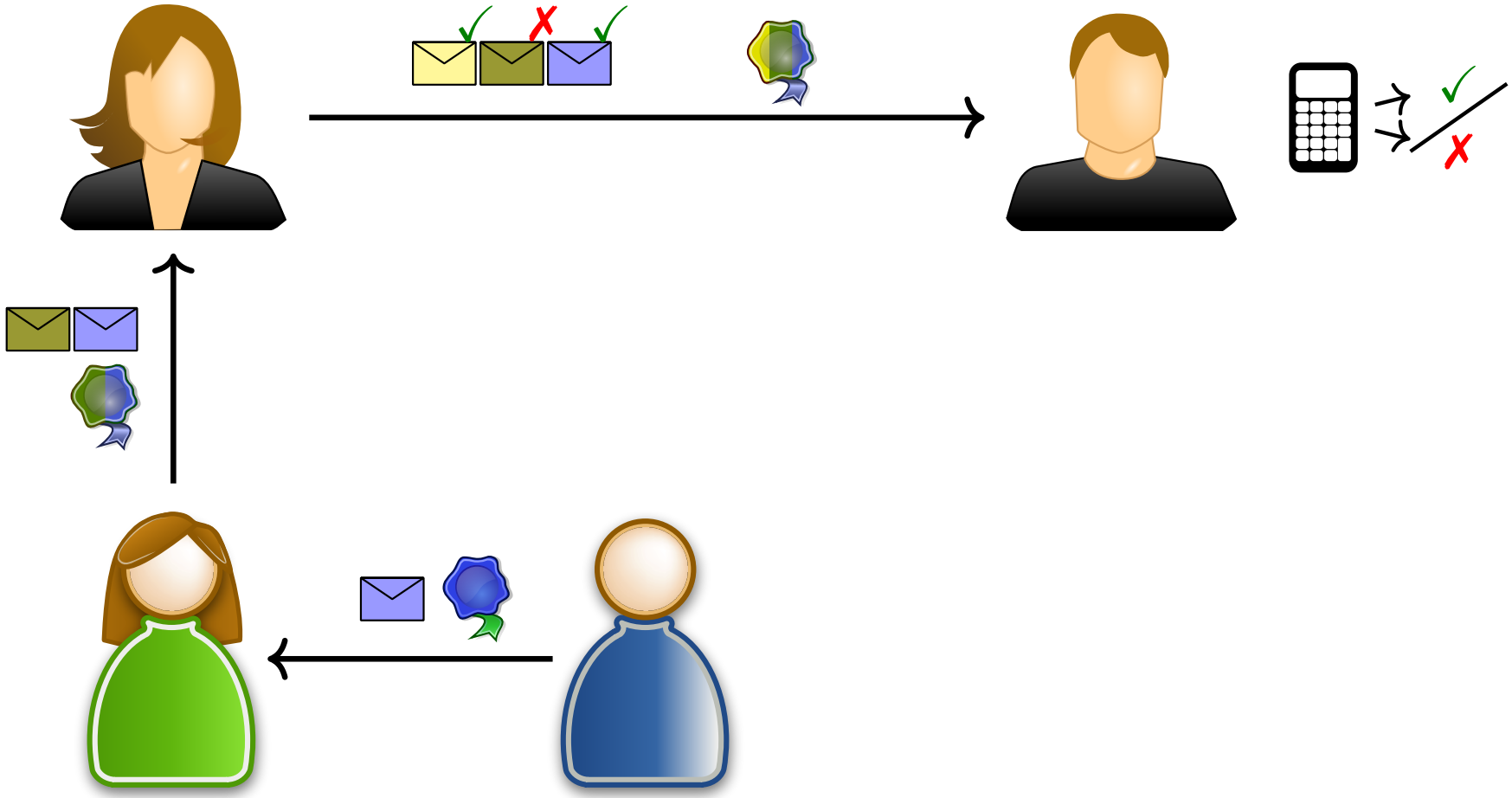
Lysyanskaya et al. [Lys+04]



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Relaxed: Sequential Aggregate Signatures

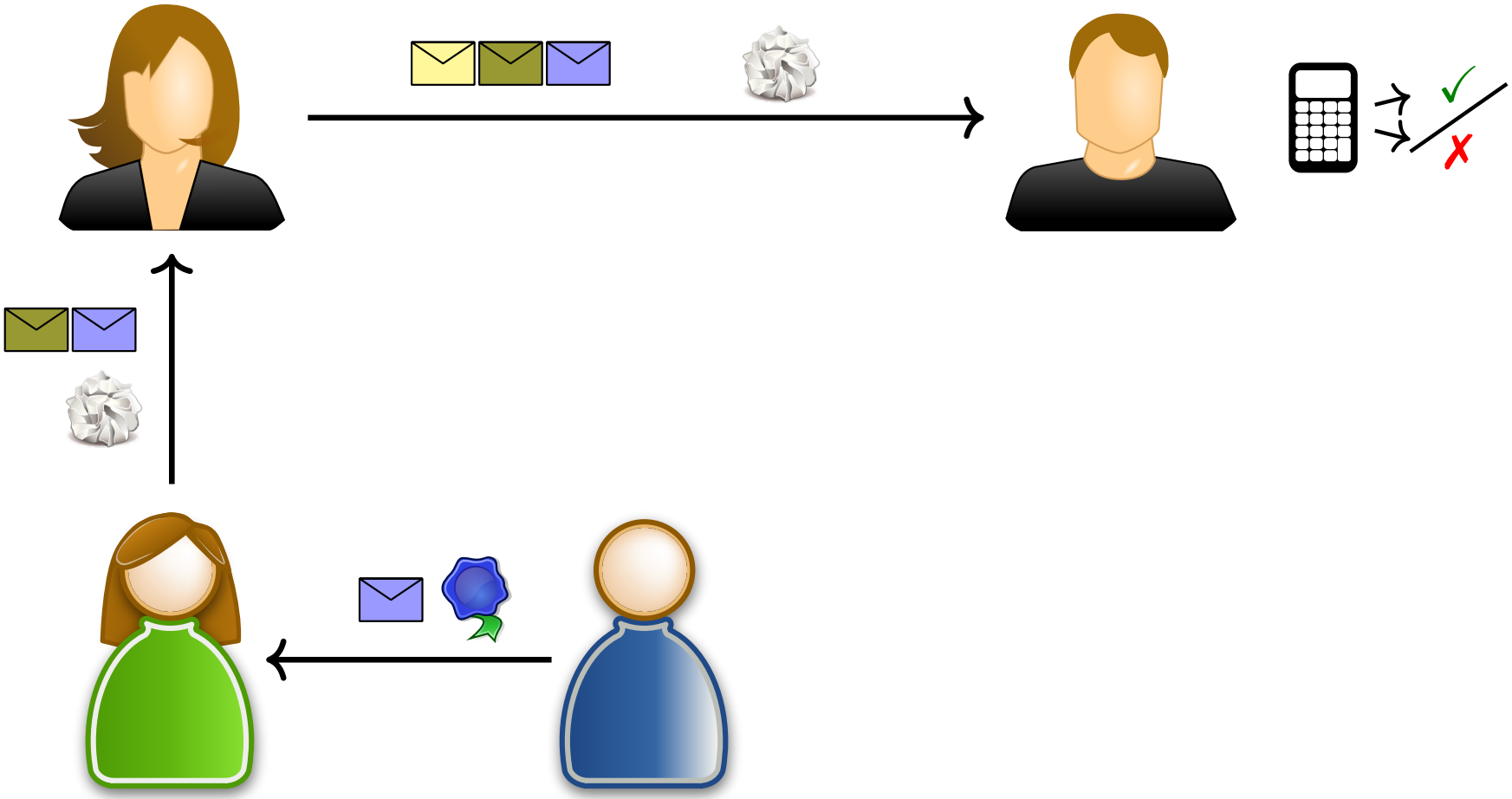
Lysyanskaya et al. [Lys+04], New: Adapt Robustness



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Relaxed: Sequential Aggregate Signatures

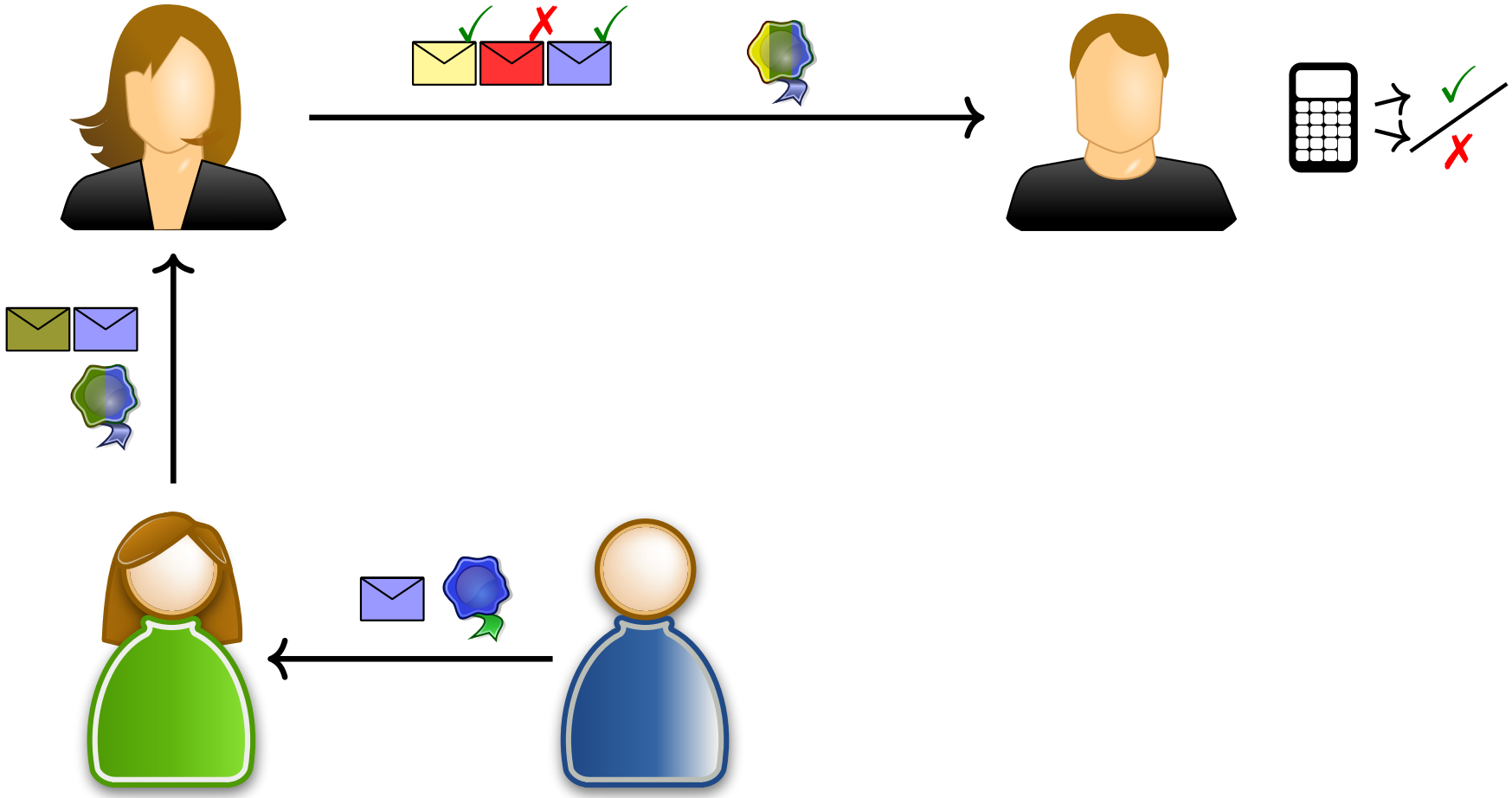
Lysyanskaya et al. [Lys+04], New: Adapt Robustness



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Relaxed: Sequential Aggregate Signatures

Lysyanskaya et al. [Lys+04], New: Adapt Robustness



Images: CC-BY-SA-3.0 Unported by RRZE, CC-0 by Tango Desktop Project, dagobert83, computating, cinemacookie. Crumpled paper by andegro4ka.

Robustness/Fault-Tolerance

Adapted from Hartung et al. [Har+16]

- The verification algorithm outputs a list of valid messages.
- A message seq. C is regular for σ, if σ is a generated sig for C.

Definition: Fault-Tolerance (informal)

A sequential aggregate signature scheme is d-fault-tolerant, iff

- for any C, σ as above, any C' with distance $\leq d$ from C (of possibly different length)
- Verify(C', σ) outputs (at least) all valid messages from C'.

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \quad \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix} \begin{matrix} \text{✗} \\ \\ \text{✗} \\ \\ \end{matrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) & \text{✗} \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) & \text{✓} \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) & \text{✗} \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) & \text{✓} \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) & \text{✗} \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) & \text{✓} \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) & \text{✗} \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) & \text{✓} \end{pmatrix}$$

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) & \text{✗} \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) & \text{✓} \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) & \text{✗} \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) & \text{✓} \end{pmatrix}$$

Why does this work?

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) & \text{✗} \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) & \text{✓} \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) & \text{✗} \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) & \text{✓} \end{pmatrix}$$

Why does this work?
No single column “covers” any other column.

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) & \text{✗} \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) & \text{✓} \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) & \text{✗} \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) & \text{✓} \end{pmatrix}$$

Why does this work?
No single column “covers” any other column. (But 2 do!)

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix} \begin{matrix} \text{✗} \\ \text{✗} \\ \text{✗} \\ \text{✗} \end{matrix}$$

Why does this work?
No single column “covers” any other column. (But 2 do!)

Review: Approach from [Har+16]

- signature $\hat{=}$ vector of signatures of underlying scheme
- columns $\hat{=}$ individual signatures

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \sigma_4 & 0 & \sigma_6 \\ \sigma_1 & \sigma_2 & 0 & 0 & \sigma_5 & 0 \\ 0 & \sigma_2 & \sigma_3 & \sigma_4 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & \sigma_5 & \sigma_6 \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{Agg}(\sigma_1, \sigma_4, \sigma_6) \\ \mathrm{Agg}(\sigma_1, \sigma_2, \sigma_5) \\ \mathrm{Agg}(\sigma_2, \sigma_3, \sigma_4) \\ \mathrm{Agg}(\sigma_3, \sigma_5, \sigma_6) \end{pmatrix} \begin{matrix} \text{✗} \\ \text{✗} \\ \text{✗} \\ \text{✗} \end{matrix}$$

Why does this work?
No single column “covers” any other column. (But 2 do!)

Incidence matrices of cover-free families [KRS99] allow for more (d) invalid signatures. Caveat: compression ratio is $\Omega(\log_2(n))$

Security Experiment

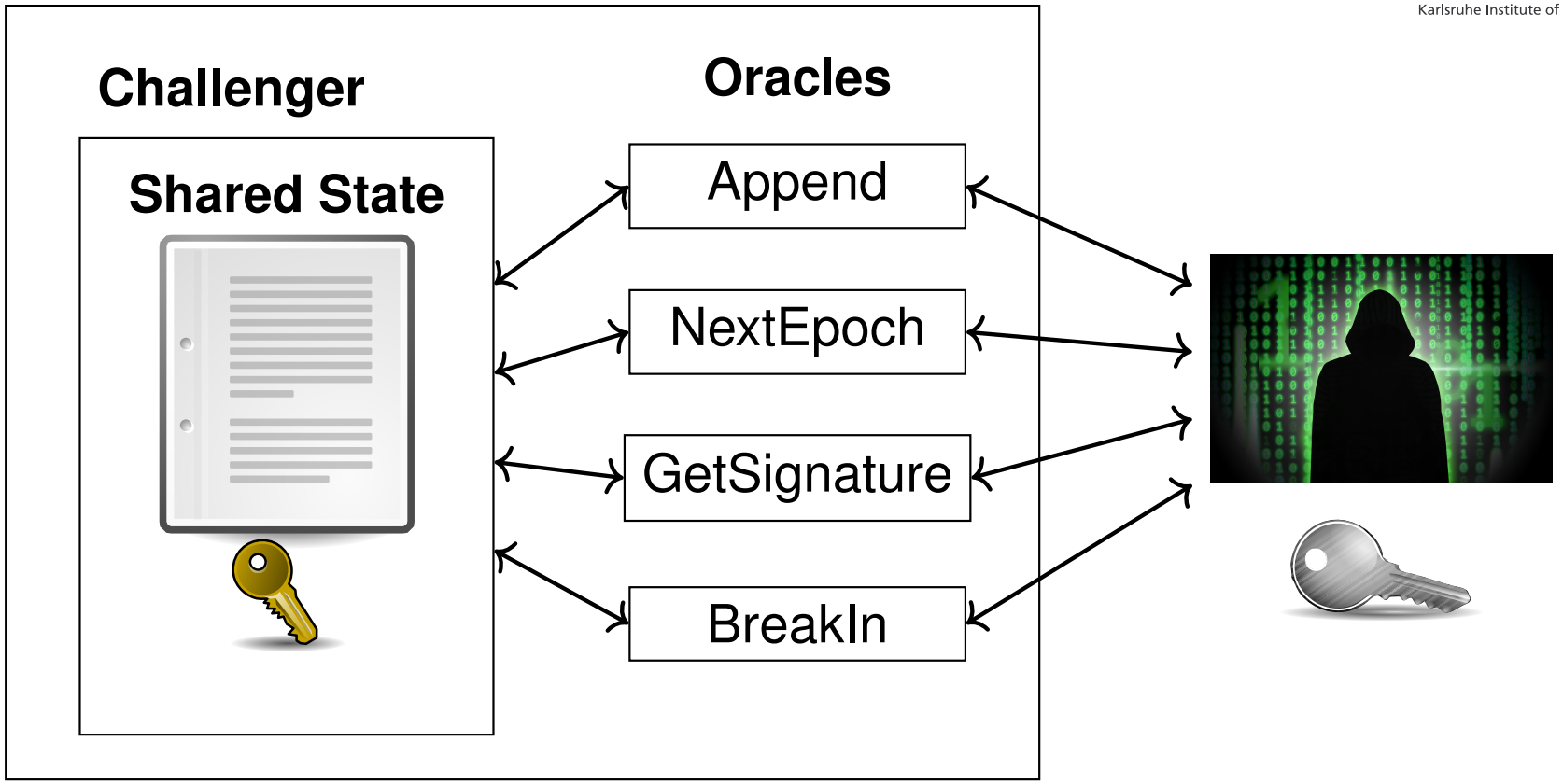
Karlsruhe Institute of Technology

Challenger

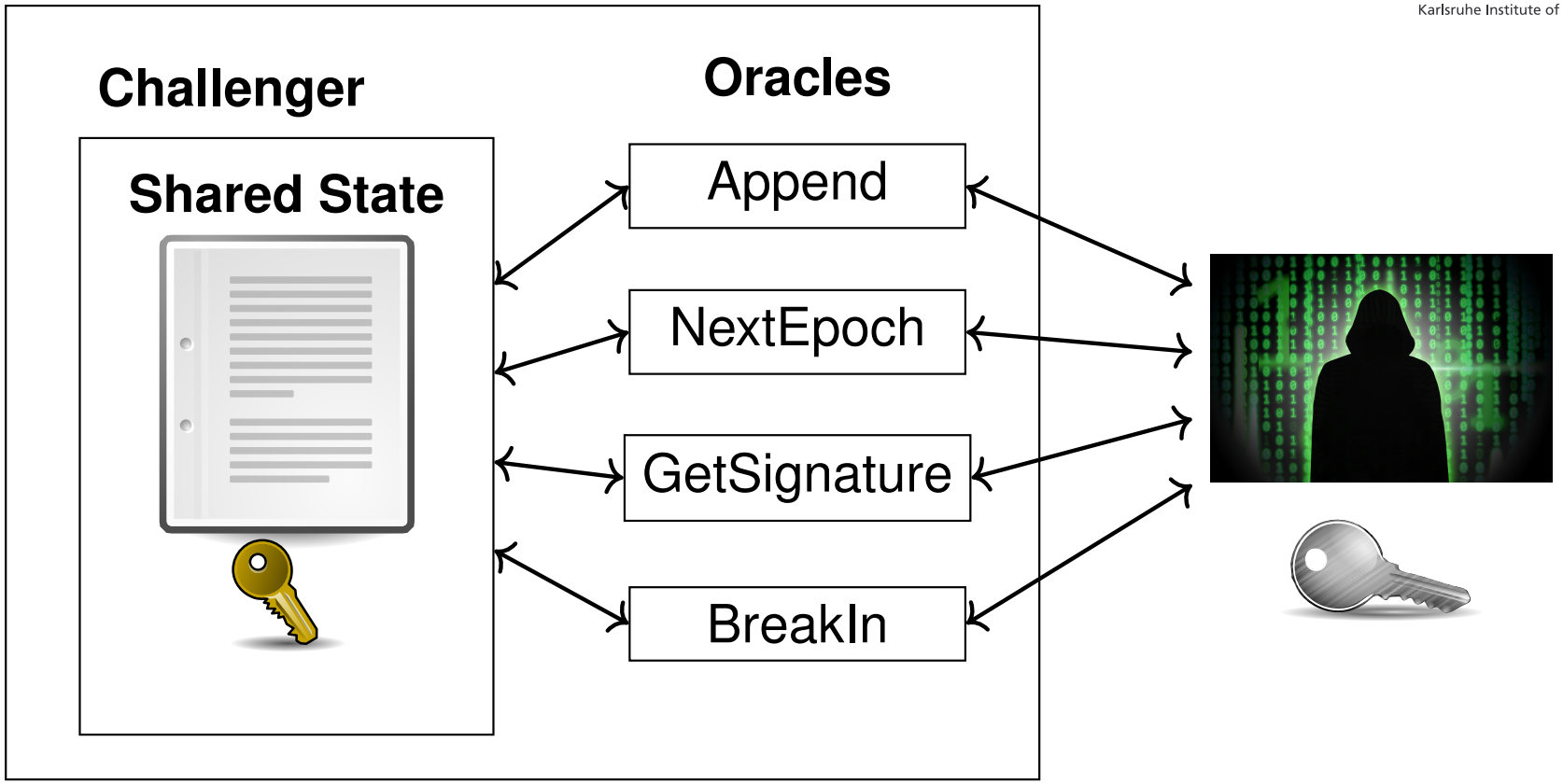
Oracles

Shared State

Security Experiment



Security Experiment



A forgery σ^* is error-free, i.e. $\text{VerifyLog}(C^*, \sigma^*) \cap \{\perp_{\text{sig}}, \perp_{\text{len}}, \perp_{\text{em}}\} = \emptyset$.

Trivial Forgeries: log states from signatures requested from $\mathcal{A}$'s GetSignature oracle or (if there has been a break in) any continuations of these using signatures from/after the break in epoch. (excl. empty log)

Our generic construction

Ingredients:

- A forward-secure sequentially aggregate signature scheme FSSAS (for creating a aggregate signature over the log across epochs)
- A forward-secure signature scheme FS (as a signature on the current log length, to be securely erased after append)
- A cover-free family CFF (for robustness)

Theorem (Informal)

Our log scheme is secure, if FSSAS and FS are secure respectively.

Proof Technique: We split the event space of what happens (whether the length signature or the log aggregate signature was forged) in two and reduce to the respective security experiments.

Implementation Benchmark

BGLS-based FSSAS (160bit), Bellare-Miner-based FS (80bit)

Algorithm	**Parameter**	ℓ	**avg** [ms]	**STD** [ms]
KeyGen	$T = 10\,000$		38 053	241
Update	$T = 10\,000$		18.6	0.033
AggSign + Update	$n = 1000$	10	67.5	0.014
	$n = 1000$	100	60.4	0.048
	$n = 1000$	1000	59.7	0.019
Verify	$n = 1000$	10	271	2.05
	$n = 1000$	100	227	1.87
	$n = 1000$	1000	22.5	0.15

On a laptop w/ Intel Core i5-2430M CPU with 2.4 GHz, equipped with 5.7 GiB of RAM. Using Shoups NTL library for BM-FSS, PBC library for BGLS-FS-SAS.

T: max. nr of epochs supported

n: nr. of messages

ℓ: nr. of messages per epoch

avg: average runtime for algorithm (STD: standard deviation)

Summary/Conclusion

- Secure logging is important. Provable Security also.
- Secure logging is hard. Mostly because of truncation.

Contribution

- We adapt the notion of fault-tolerance from [Har+16] to sequential aggregate signatures, adapt their generic construction and prove its security and fault-tolerance,
- we give a strong security notion capturing truncation attacks,
- we give a generic space-efficient construction of a publicly-verifiable robust secure logging scheme, which has a tight reduction,
- we benchmark an implementation of our scheme.

Future work: Find a forward-secure sequential aggregate signature scheme without pairings (i.e. that is not already fully aggregatable).

References: I

M. Bellare and S. K. Miner. “A Forward-Secure Digital Signature Scheme”. In: CRYPTO 1999. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, 1999, pp. 431–448.

D. Boneh, C. Gentry, B. Lynn, and H. Shacham. “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps”. In: EUROCRYPT 2003. Ed. by E. Biham. Vol. 2656. LNCS. Springer, 2003, pp. 416–432.

G. Hartung, B. Kaidel, A. Koch, J. Koch, and A. Rupp. “Fault-Tolerant Aggregate Signatures”. In: PKC 2016, Part I. Ed. by C. Cheng, K. Chung, G. Persiano, and B. Yang. Vol. 9614. LNCS. Springer, 2016, pp. 331–356.

References: II

R. Kumar, S. Rajagopalan, and A. Sahai. “Coding Constructions for Blacklisting Problems without Computational Assumptions”. In: CRYPTO 1999. Ed. by M. J. Wiener. Vol. 1666. LNCS. Springer, Aug. 1999, pp. 609–623.

A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. “Sequential Aggregate Signatures from Trapdoor Permutations”. In: EUROCRYPT 2004. Ed. by C. Cachin and J. Camenisch. Vol. 3027. LNCS. Springer, 2004, pp. 74–90.

D. Ma and G. Tsudik. “A New Approach to Secure Logging”. In: Data and Applications Security XXII, IFIP WG 11.3 Working Conference on Data and Applications Security. Ed. by V. Atluri. Vol. 5094. LNCS. Springer, 2008, pp. 48–63.