Anonymously Publishing Liveness Signals with Plausible Deniability

Michael Sonntag, René Mayrhofer, Stefan Rass

JOHANNES KEPLER
UNIVERSITÄT LINZ

René Mayrhofer
Institute of Networks and Security

INSTITUTE
OF NETWORKS
AND SECURITY

Problem statement

Dall-E with prompt “A picture of a brave woman blowing a whistle while holding up a document folder in film noir style”

Problem statement

- Persons might want to prove ("**Prover**") to others ("**Verifier**") that they are still alive & well ("**Signal**" received)
  - E.g. whistleblowers; to keep a "security package" stashed with third parties from being published
- If you want/need such a scheme, you implicitly want to remain **anonymous** – and in case of suspicion **be able to deny** it
  - "I am not a whistleblower" (who would be sent to jail)
  - "I am not a 'verifier' with a package" (who would have to relinquish the package – and maybe go to jail too – participation/help/…)
  - Even if the other party is discovered, all devices are obtained & analyzed, and the person (made to) cooperate, you should still be able to disclaim any participation
- We developed and implemented a protocol to support this:
  - Proving "**liveness**" + **plausible deniability**
  - Not included: data communication, security package etc.

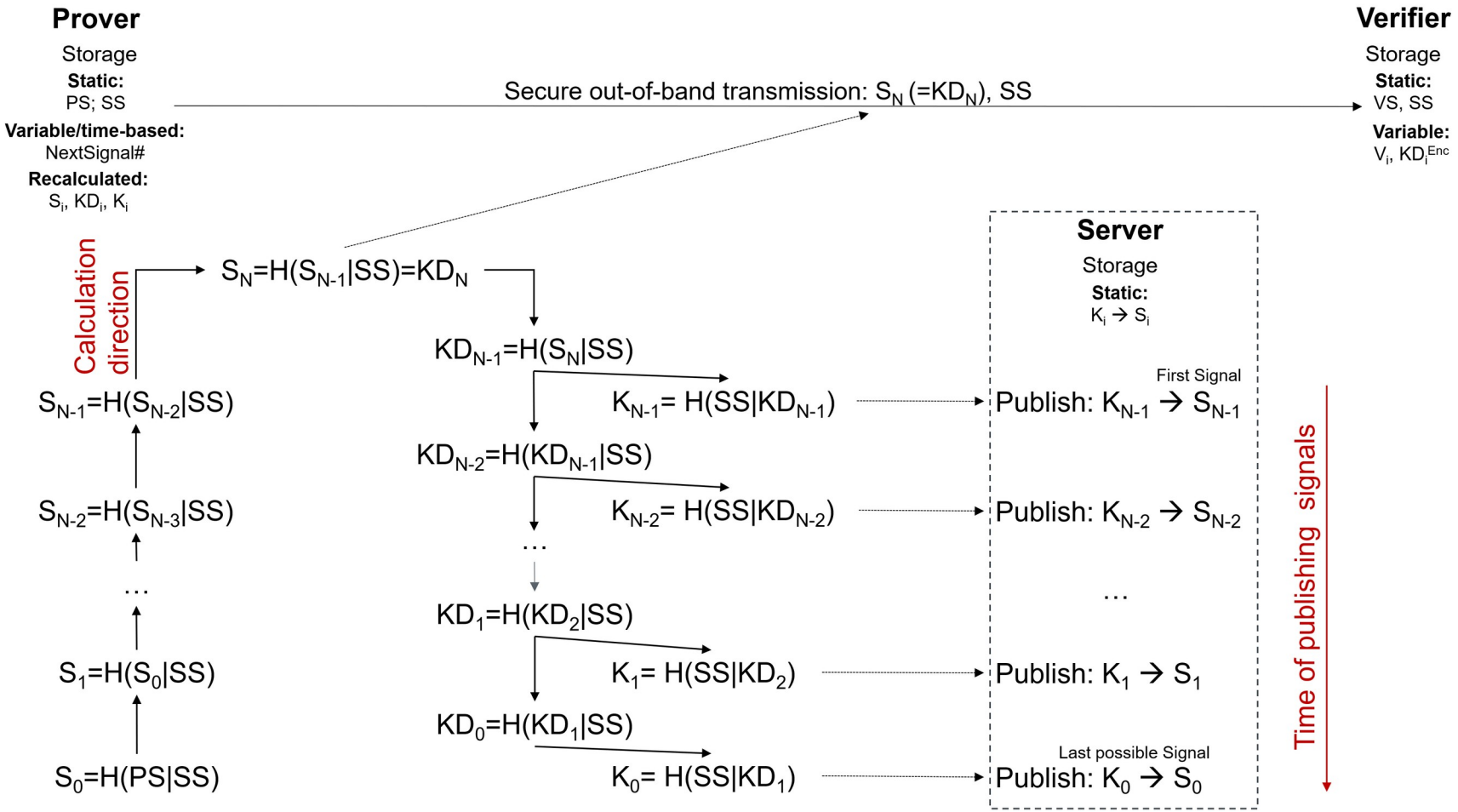
Solution outline [1]

- Store “Signal” on a third-party server, so communication is completely **asynchronous** → no correlation attacks
  - Communication should be explainable as “normal usage”, too
- Use Tor & **Onion services** to hide participants
  - Large foreign public Onion services (= Signal storage) preferable
  - Querying non-existing signals: random generation
- Roles are **symmetrical** regarding stored data
  - Each one can claim to be the other (if desirable)
- **No identical data** stored at participants
  - Exception: a single shared secret kept in (human) memory
  - Prover: nothing related found at Verifier, even if all secrets (incl. shared one) are disclosed correctly
  - Verifier: vice versa
  - Both: Lying about anything provides valid values indistinguishable from those based on correct disclosure

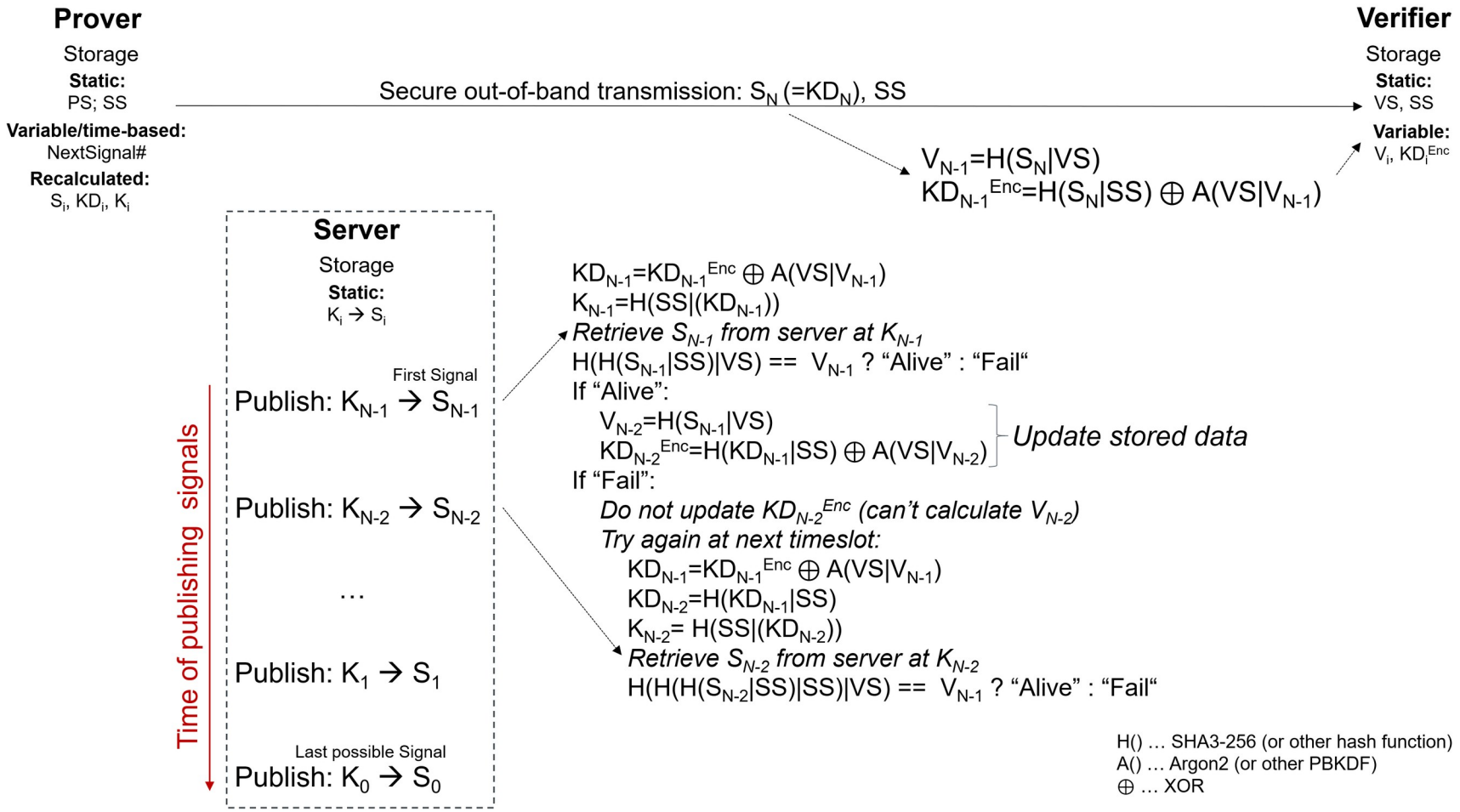
Solution outline [2]

- No danger from attackers for storage server
  - Proof of Work for querying and submission (DoS prevention)
    - No DoS regarding signals or protocol; requires only limited storage
  - No registration, payment, etc. needed
- A “key” is used to distinguish multiple provers on a single server
  - Derived from the same values; properties as before
- Signals may be missed: participants can calculate/verify forward, but not backwards
  - Based on a hash chain → no reversing (computational limits)
  - The prover can stop calculation early to create “old” keys/signals, but that doesn’t help with identifying/proving a verifier
  - After a freely set number of missed signals the verifier considers the prover “dead” → can delete data, publish security package, ...
    - And should stop verification attempts!

Prover side


Prover
Storage
Static:
PS; SS
Variable/time-based:
NextSignal#
Recalculated:
S_i, KD_i, K_i
Secure out-of-band transmission: S_N (=KD_N), SS
Verifier
Storage
Static:
VS, SS
Variable:
V_i, KD_i^{Enc}
Server
Storage
Static:
$K_i \rightarrow S_i$
Calculation direction
$S_N=H(S_{N-1}|SS)=KD_N$
$KD_{N-1}=H(S_N|SS)$
$S_{N-1}=H(S_{N-2}|SS)$
$K_{N-1}= H(SS|KD_{N-1})$
First Signal
Publish: $K_{N-1} \rightarrow S_{N-1}$
$KD_{N-2}=H(KD_{N-1}|SS)$
$S_{N-2}=H(S_{N-3}|SS)$
$K_{N-2}= H(SS|KD_{N-2})$
Publish: $K_{N-2} \rightarrow S_{N-2}$
$KD_1=H(KD_2|SS)$
$S_1=H(S_0|SS)$
$K_1= H(SS|KD_2)$
Publish: $K_1 \rightarrow S_1$
$KD_0=H(KD_1|SS)$
$S_0=H(PS|SS)$
$K_0= H(SS|KD_1)$
Last possible Signal
Publish: $K_0 \rightarrow S_0$
Time of publishing signals

Verifier side

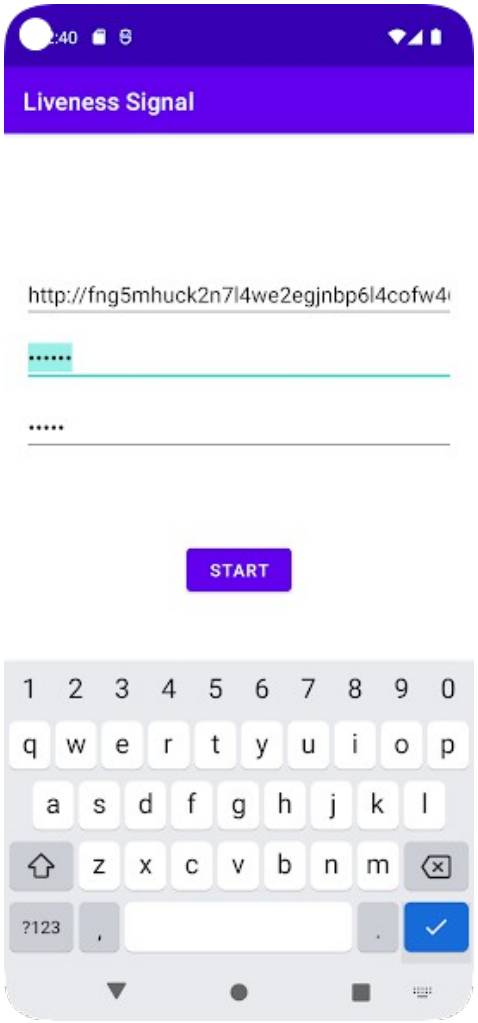


Data “stored” by participants

- Prover & Verifier:
  - Onion address of storage server(s): public site, human memory, …
  - Shared secret (human memory only)
- Prover:
  - Prover secret (human memory only) for signal/key generation
  - Number of the next signal
    - Or some method of deriving it, e.g. starting time + current date/time
  - Arbitrary data looking like current key generation/verification data
- Verifier:
  - Verifier secret (human memory only)
  - Current key generation data
    - Encrypted via XOR with data derived from verifier secret and verification data during storage & ratcheted forward after each sending
  - Verification data for verifying the next signal value
- Onion service operator: Map[Key → Signal]

Single hash value each

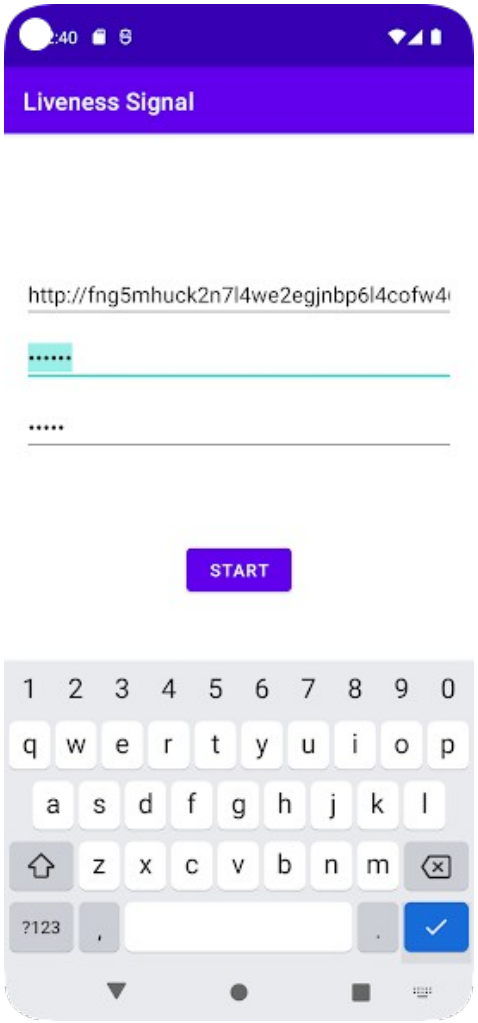
Exemplary implementation

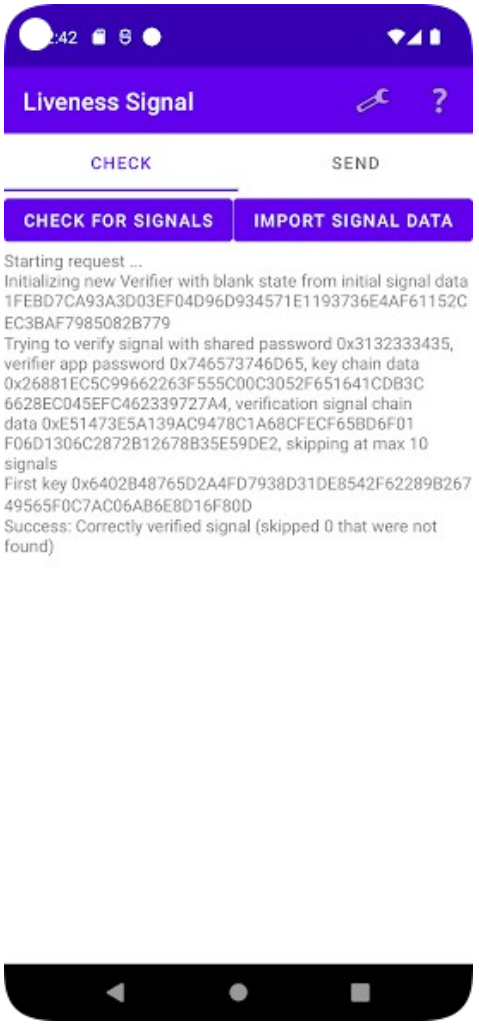


Exemplary implementation

- Same Android app for Prover & Verifier:
  - Default onion address for storage server (run by INS at JKU)
  - App secret specific to each user, used for local storage encryption
  - Shared secret (human memory only) for signal creation+verification

Exemplary implementation





Exemplary implementation

- Same Android app for Prover & Verifier:
  - Default onion address for storage server (run by INS at JKU)
  - App secret specific to each user, used for local storage encryption
  - Shared secret (human memory only) for signal creation+verification
- Synchronizing Prover & Verifier:
  - One-time initial synchronization, assisted by displaying a QRcode at Prover and scanning with Verifier
  - After that first synchronization, completely asynchronous communication through the Onion service
- Core cryptographic protocol implemented in Java-only library
  - Minor dependencies (mostly logging)
  - Can be easily used in other apps, e.g. standard news organizations apps with integrated messaging functionality

Plausible deniability achieved? [1]

- Prover cooperates and discloses prover and shared secret
  - Future keys and signals can be calculated
    → Prover can be impersonated
  - None of that data is found at the verifier on any device, neither the shared secret nor any of the future key or verification data
  - Calculating older keys does not help, as servers do not store when/whether the data was retrieved – and would they, this would not help either with identifying/proving a Verifier because of Tor
  - Lying about the shared secret produces valid values that can be stored (but will not validate); previous ones (allegedly published in the past) are no longer stored by the server and enumeration by attacker in advance is impossible
  - Correlation attacks can be performed, but require cooperation of the storage server → pre-calculate the key and wait for check(s)

Plausible deniability achieved? [2]

- Prover can claim to be a verifier: With an invented (or correct) shared secret signals can be successfully retrieved, but none will validate
  - Some delay required to convincingly tell “prover is already dead”
  - Old signals cannot be generated, so it is impossible to prove that there never was a valid signal
- Verifier: Situation is symmetric
  - Verifier can be impersonated if disclosing all values
    - Verifying liveness becomes possible for the attacker
  - No help identifying/proving the Prover
    - No matching data found there; no access to previous keys or signals as this would require reversing the hash function
  - Verifier could claim to be a prover: that no one verifies this could only be proven together with the storage server & if quick
    - Or prover would already consider him dead and checks no longer

Summary

- We provide a scheme to prove a “recent activity” by “someone knowing a shared secret”
  - But without the ability for attackers to identify any participant, knowing such a scheme is going on, and even if all (other) participants cooperate fully, the last one can still deny involvement
    - Or claim a different role, if desirable
- Open problems:
  - Separate app needed: integration (tiny part) into a widely-used app would remove this sign of participation
    - Alternative: Download JavaScript from trusted website and calculate locally; difficult to verify this is secure (unchanged code); requires lots of trust in the site
  - Third party needed for storage
    - Load is low: practically no computational effort required
    - Storage: 1 attacker doing 24/7 nothing else: ≈ 15 MB storage

JOHANNES KEPLER
UNIVERSITÄT LINZ

INSTITUTE
OF NETWORKS
AND SECURITY

https://www.ins.jku.at

THANK YOU FOR YOUR ATTENTION!

Questions?

Michael Sonntag
michael.sonntag@ins.jku.at
+43 (732) 2468 - 4137
S3 235 (Science park 3, 2nd floor)

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
www.jku.at