Randomness Improvements for Security Protocols

draft-cremers-cfrg-randomness-improvements

Cas Cremers (cas.cremers@cs.ox.ac.uk)
Luke Garratt (luke.garratt@cs.ox.ac.uk)
Stanislav Smyshlyaev (svs@cryptopro.ru)
Nick Sullivan (nick@cloudflare.com)
Christopher A. Wood (cawood@apple.com)

CFRG
IETF 101, March 2018, London

Background

PRNGs can break or contain design flaws

- Debian bug [1]: PRNG seeding process broken by removing crucial mixing step

```
MD_Update(&m,buf,j); /* pre-seed */
[ .. ]
MD_Update(&m,buf,j); /* post-seed */
```

- Dual_EC [2,3]: Backdoored design could be exploited by TLS implementations, especially when Extended Random is used

[1] Diff of /openssl/trunk/rand/md_rand.c
[2] On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng
[3] On the Practical Exploitability of Dual EC in TLS Implementations

Rationale

Build on NAXOS trick [4]:

- Replace raw entropy x with H(x || sk)

Defense in depth mentality

- Distinguishability guarantees reduce to secret key security for broken PRNGs

PRNG failures are localized if they occur

[4] LaMacchia, Brian et al., "Stronger Security of Authenticated Key Exchange."

Private Keys

Direct access to private keys is not always possible

- Servers may store them in HSMs
- Clients may store them in enclaves

Keys are of varying types — RSA, EC-based

Commonality: used to *compute* private key operation (signature) at some point during protocol execution

Randomness Wrapper

$G(x)$ Generate x random bytes

$\mathsf{PRF}(k, x)$ Compute PRF of input x with key k

$\mathsf{Sig}(\mathsf{sk}, m)$ Compute signature of message m using secret key sk

$\mathsf{KDF}(x) = \mathsf{HKDFExtract}(\perp, x)$

$$\boxed{\mathsf{PRF}(\mathsf{KDF}(G(x) || H(\mathsf{Sig}(\mathsf{sk}, \mathsf{tag1}))), \mathsf{tag2})}$$

Extractor

Expander

Details

$$\boxed{\mathsf{PRF}(\mathsf{KDF}(G(x)||H(\mathsf{Sig}(\mathsf{sk},\mathsf{tag1}))),\mathsf{tag2})}$$

Tags prevent collisions across private key operations:

- tag1: Constant string bound to device and protocol
- tag2: Dynamic string — timestamp, counter, etc.

Signature $\mathsf{Sig}(\mathsf{sk}, m)$ MUST NOT be exposed

Signature algorithm SHOULD be deterministic

Criticism

- Why bother? PRNGs are easy to get right…
- Why use your private key for something unintended?
- …

Open Issues & Next Steps

Not a drop-in replacement for /dev/random

- Analyzing AES-based extraction wrappers, similar to (expired) draft-agl-ckdf

Experiment with existing implementations

- Simple Go implementation available at [5]
- BearSSL and NSS, among others, include user-space PRNG implementations

[5] https://github.com/chris-wood/draft-cremers-cfrg-randomness-improvements/blob/master/poc/augmented_random.go