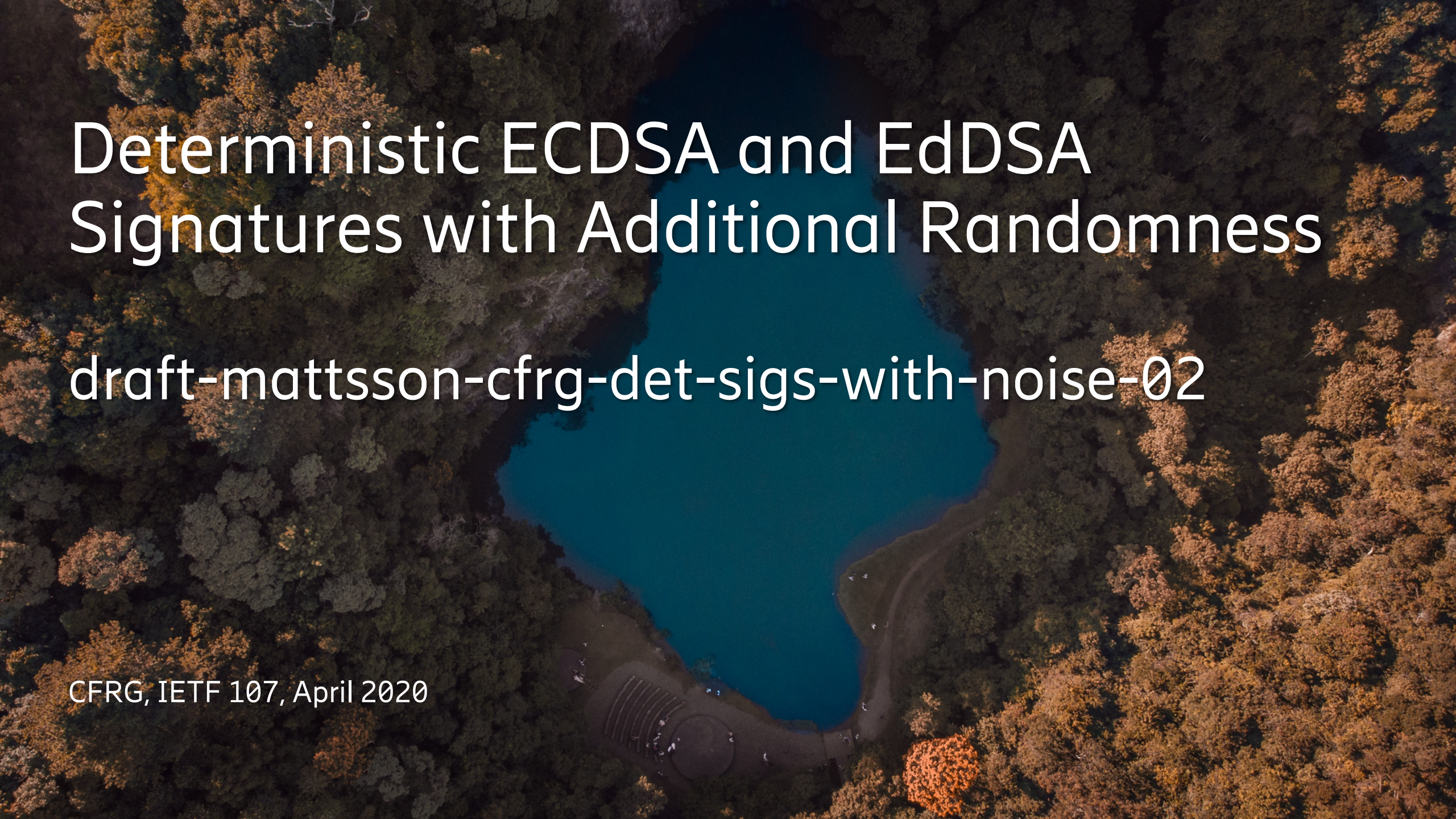

Deterministic ECDSA and EdDSA Signatures with Additional Randomness

draft-mattsson-cfrg-det-sigs-with-noise-02

CFRG, IETF 107, April 2020

Randomized ECDSA and Deterministic ECDSA and EdDSA Signatures

- Biases in the random number generation for randomized ECDSA [FIPS-186-4] may have catastrophic effects such as compromise of the private key.
- A large number of RFCs are currently recommending deterministic ECC signatures (Deterministic ECDSA and EdDSA) [RFC8037] [RFC8080] [RFC8152] [RFC8225] [RFC8387] [RFC8410] [RFC8411] [RFC8419] [RFC8420] [RFC8422] [RFC8446] [RFC8463] [RFC8550] [RFC8591] [RFC8624] [RFC8208] [RFC8608].
- Recent research show that key compromise from side-channel and fault injection attacks on deterministic ECC signatures are practically feasibly in some environments. Especially in IoT deployments. [SH16] [BP16] [RP17] [ABFJLM17] [SBBDS17] [PSSLR17] [SB18] [WPB19] [AOTZ19] [FG19]
- US NIST is discussing these threats in [Draft-186-5] and German BSI has written and co-authored several publications on the topic.

Industrial IoT device by Kit Teco
https://www.flickr.com/photos/teco_kit/with/23908928999/

Deterministic ECDSA and EdDSA Signatures with Additional Randomness

- One countermeasure to side-channel and fault injection attacks recommended by [RP17] [ABFJLM17] [SBBDS17] [PSSLR17] [SB18] [AOTZ19] [FG19] and implemented in [XEdDSA] [libSodium] [libHydrogen] is to re-introduce some additional randomness to the otherwise deterministic generation of the per-message secret number. Also known as hedged signatures.
  - Simple and well understood.
  - Works for both ECDSA and EdDSA.
  - Minor modifications to signing, none to verification.
  - Does not increase the number of point multiplications.
  - With weak randomness, hedged signatures are still as secure as deterministic signatures.

Suggested Updates to RFC 8032 and RFC 6979

- Many current and future IoT deployments will use ECC signatures, e.g. TLS 1.3 [RFC8446] and COSE [RFC8152].
- IETF / IRTF should quickly publish updated recommendation for use of ECC signatures in deployments where side-channel and fault injection attacks are a concern.
- Is CFRG the right place?
- Version -02 updates the construction and terminology based on suggestions from Quynh Dang, Uri Blumenthal, and Tony Arcieri.
  - Concatenation with Z instead of XOR.
  - Random data Z inserted before the private key.
  - Use of zero padding 000… to separate key and message.
- Recommended for some or all deployments?
- Any other improvements to the construction?
  - Length of random data Z?

2. Updates to RFC 8032 (EdDSA)

For Ed25519ph, Ed25519ctx, and Ed25519: In deployments where side-channel and fault injection attacks are a concern, the following step is RECOMMENDED instead of step (2) in Section 5.1.6 of [RFC8032]:

2. Compute SHA-512(dom2(F, C) || Z || prefix || 000... || PH(M)), where M is the message to be signed, Z is 32 octets of random data, the number of zeroes 000... is chosen so that the length of (dom2(F, C) || Z || prefix || 000...) is 1024 bytes. Interpret the 64-octet digest as a little-endian integer r.

3. Updates to RFC 6979 (Deterministic ECDSA)

For Deterministic ECDSA: In existing ECDSA deployments where side-channel and fault injection attacks are a concern, the following steps are RECOMMENDED instead of steps (d) and (f) in Section 3.2 of [RFC6979]:

d. Set:

K = HMAC_K(V || 0x00 || Z || int2octets(x) || 000... || bits2octets(h1)) where '||' denotes concatenation. In other words, we compute HMAC with key K, over the concatenation of the following, in order: the current value of V, a sequence of eight bits of value 0, random data Z (of the same length as int2octets(x)), the encoding of the (EC)DSA private key x, a sequence of zero bits 000... chosen so that the length of (V || 0x00 || Z || int2octets(x) || 000...) is equal to the block size of the hash function