Evaluation of secp256k1 as Popular Alternative Curve

Christopher Allen, Principal Architect / Blockstream
CFRG Interim Meeting, Paris — April 30, 2017

What is secp256k1?

Variant of ECDSA:

- ECDSA was created by NSA and is defined in NIST 186-3 DSS Standard, and uses the elliptic curve secp256r1
- secp256k1 addressed concerns by cryptographic community as to possible hidden parameters in the ECDSA random coefficient
- Simpler structure, fewer choices, thus less ways for a malicious party to introduce vulnerabilities

Improvements in secp256k1:

- Significant performance improvements over NIST-based ECDSA, which is currently allowed for standards
- Uses a Koblitz-like curve for efficient computation, is often ~30% faster for verification than ECDSA
- Currently ~30% slower than Ed25519-donna for signing, but anticipating some future speed improvements

Why is secp256k1 important?

De-facto standard for blockchains

- Used by Bitcoin, Ethereum, Zcash and many other blockchains
- In Bitcoin alone, ~260K transactions a day for ~$390M volume per day
- Protecting $25B+ markets!
- "Largest Bug Bounty in the World!"
- May not be most current work in elliptic curve, but "good enough"

Significant usage

- In active use since 2009
- Multiple interoperable implementations
- Multiple languages: libsecp256k1 (C), Bouncy Castle (Java VM), Crypto++ (C++), secp256k1-go (Go), tiny-secp256k1 (Rust), elliptic-curve-js (Javascript)

libsecp256k1: fast, strong, well-reviewed, well-tested

Most used implementation: libsecp256k1 (C)

- Fast: validation of signatures increased 5x over original OpenSSL
- Significant review and high test coverage
- Hand verifiable proof of correctness for the field multiplication algorithm
- Computer verified proof of correctness for group addition formulae
- Special compilable mode that changes a constant to end up with a very small group, and exhaustive tests that all assumptions remain true
- Test cases for the scalar code that were extracted from a set of 1 trillion randomly generated tests which give very high coverage, and work in progress to algebraically derive cases that trigger the (nearly) unreachable remaining ones

Why not use a more current standard?

Why not use Ed25519 or a more recent curve?

- More modern curves did not exist or were not well studied when first blockchains began
- Due to stability required by consensus protocols and financial code, established blockchains can't easily switch to other curves

- Ed25519 is non-linear, thus there have been no standards for HD (Hierarchical Deterministic) Key derivation used by most blockchains

Why allow secp256k1 for Standards?

Blockchain standards are local and ad-hoc

- Many blockchain communities are already using secp256k1 with JWT. However, all are non-conformant to published standard
- We can improve security by making secp256k1-based implementations conform to standards
- Support of secp256k1 brings blockchain communities into standards efforts

Where to be used?

- For interoperability reasons, greatest need is in W3C to support secp256k1 in Web Payments & Verifiable Claims Working Groups
- These W3C groups use IETF JOSE standards for such as JWS for JSON Signing and Encryption.
- Curve not be requested for all existing standards, for instance no requests to secp256k1 add to TLS or SSH.

What do we need to do get CFRG to evaluate and approve secp256k1 for optional use in IETF standards?

•••

ChristopherA@LifeWithAlacrity.com
ChristopherA@Blockstream.com