Sign on the Dotted Line

FIPS 186: The Digital Signature Standard

Dustin Moody (NIST)

NIST Crypto Reading Club
October 30, 2024

Outline

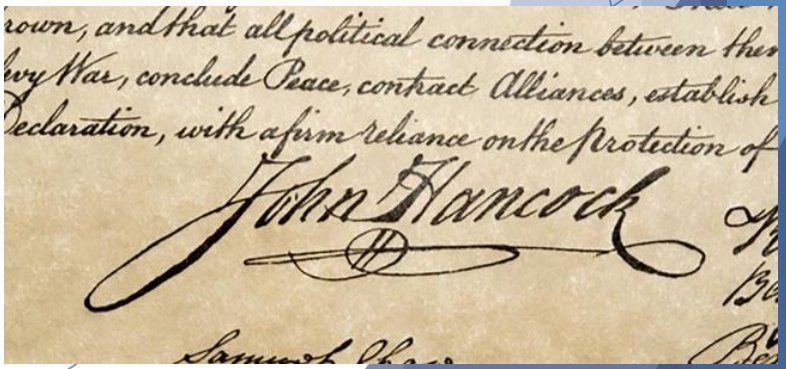
- What is a digital signature?
- History of FIPS 186
- What's in FIPS 186-5
  - Signatures
    - (DSA)
    - RSA
    - ECDSA
    - EdDSA
  - Other stuff
- Related, but not in FIPS 186
  - SP 800-186
  - PQC signatures

Historical Signatures

- 3000 B.C.E. – Cuneiform tablets w/ signatures

- Middle Ages – Signet rings and wax seals

- 1776 – Declaration of Independence



Electronic Signatures

- 1869 – Howley v Whipple case
  - “It makes no difference whether operator writes with a steel pen an inch long attached to an ordinary penholder, or whether his pen be a copper wire a thousand miles long. Nor does it make any difference that in one case common record ink is used, while in another case a more subtle fluid, known as electricity, performs the same office.”

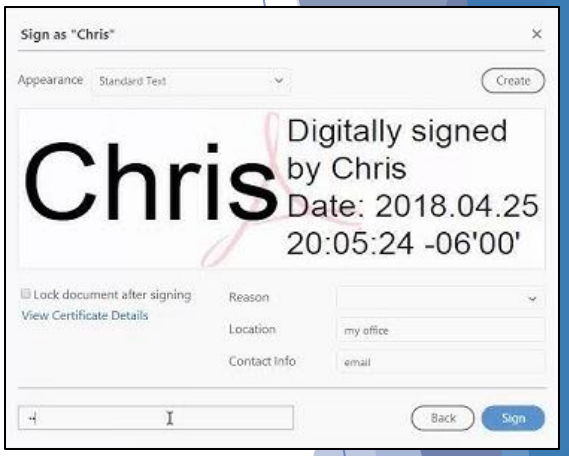
    New Hampshire Supreme Court

- 1988 – Lotus Notes 1.0
  - 1st widely marketed software to use digital signatures (RSA)

- 2000 – ESIGN law signed
  - Digital signatures legally binding

Digital Signatures

- Digital signatures are cryptographic schemes which can provide
  - Authentication
  - Integrity
  - Non-repudiation



- Typically consists of 3 algorithms
  - **Key Generation** – create public/private key pair
  - **Sign** – given a message and private key, produces a signature
  - **Verify** – given a message, a signature and a public key, either accepts or rejects the signature as authentic

The Start of Standardization

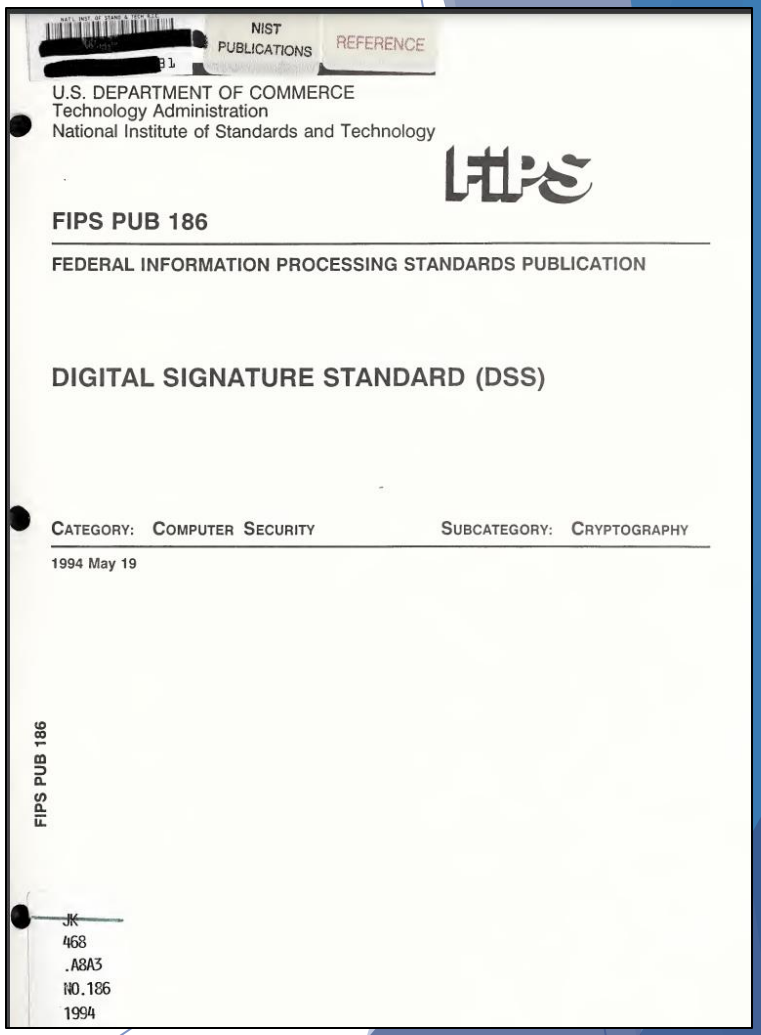
- 1978 – RSA signature algorithm published
- 1982 – Federal Register Notice soliciting digital signature algorithms
  - RSA paper is the only suggested algorithm
  - Patent issues
- 1989 – NIST/NSA Technical Working Group (TWG)
  - NIST considered several algorithms
    - RSA, already deployed in industry
    - Other algorithms from academic literature
    - NSA-designed algorithms

Public Feedback

- 1991 – NIST selected DSA for standardization
  - Draft version of FIPS 186 published
- Concerns
  - DSA selection process not public
  - Not enough cryptanalysis
  - Patent concerns
  - Parameter sizes
    - 512 bit modulus, 160 bit subgroup
  - Performance concerns
- 1992 NIST report on comments and adjudications
  - Increased parameter sizes allowed (up to 1024 bits)

The original FIPS 186

- Although favored by industry, RSA was not selected
  - Issues over exportability and patents
- FIPS 186 Published in May 1994
- 28 pages
- Specifies DSA, the Digital Signature Algorithm
  - Just 2 pages for details of DSA
  - The DSA was designed by the NSA



DSA

- KeyGen
  - Choose primes *p*, *q* with *q*|(*p-1*). Let *h* be random mod *p*
  - Compute $g = h^{(p-1)/q}$ mod *p*. Make p, q, g public
  - Private key: random *x*. Public key: $y = g^x$ mod *p*
- Sign
  - Let *k* be random mod *q*. Compute ($r = g^k$ mod *p*) mod *q*
  - Compute $s = (H(M) + xr)/k$ mod *q*
  - The signature of message *M* is (*r*,*s*)
- Verify
  - Compute $u_1 = H(m)/s$ mod *q* and $u_2 = r/s$ mod *q*
  - Compute $v = (g^{u_1} y^{u_2} \, mod \, p)$ mod *q*. Accept iff *v* = *r*

DSA

- KeyGen
  - Choose primes *p*, *q* with *q*|(*p*-1). Let *h* be random mod *p*
  - Compute $g = h^{(p-1)/q}$ mod *p*. Make p, q, g public
  - Private key: random *x*. Public key: $y = g^x$ mod *p*
- Sign
  - Let *k* be random mod *p*. Compute ($r = g^k$ mod *p*) mod *q*
  - Compute $s = (H(M) + xr)/k$ mod *q*
  - The signature of message *M* is (*r*,*s*)
- Verify
  - Compute $u_1 = H(M)/s$ mod *q* and $u_2 = r/s$ mod *q*
  - Compute $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$. Accept iff *v* = *r*

Why it works:

$$
\begin{aligned}
v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\
&= \left(g^{H(M)/s} y^{r/s} \bmod p\right) \bmod q \\
&= (g^{H(M)/s} g^{xr/s} \bmod p) \bmod q \\
&= (g^{(H(M)+xr)/s} \bmod p) \bmod q \\
&= (g^k \bmod p) \bmod q \\
&= r
\end{aligned}
$$

Security of DSA

- Necessary conditions for the security of DSA
  - The discrete log problem is hard in $< g >$
  - H is a one-way hash, and is collision-resistant
  - The per-message-secret k is unpredictable, and not re-used
- Some validation and other requirements also needed
- Best known attacks on finite field discrete logs are special versions of the function field sieve

The next 2 versions of FIPS 186

- Background
  - DSA not widely adopted
  - Interest in RSA and elliptic curve DSA schemes
- 1997 – NIST requested comments on adding new signature schemes to FIPS 186
  - Overwhelmingly positive response for both schemes
- NIST worked with ASC X9
  - X9.31 for RSA and X9.62 for ECDSA
- 1998 – FIPS 186-1, approves X9.31 (RSA)
- 2000 – FIPS 186-2, approves X9.62 (ECDSA)

Basic RSA signatures

- KeyGen
  - Let $n = pq$, where p,q are large primes
  - Let e be chosen (randomly), or set $e = 2^{16}+1=65536$
  - Compute $d = 1/e \bmod \phi(n)$, i.e. $ed \equiv 1 \, mod \, \phi(n)$
  - Public key is (n,e) and private key is (n,d)
- Sign
  - Compute $s = H(M)^d \bmod n$. The signature is s
- Verify
  - Compute $s^e \bmod n$. Accept iff this equals $H(M)$

Basic RSA signatures

- KeyGen
  - Let $n = pq$, where p,q are large primes
  - Let e be chosen (randomly), or set $e = 2^{16}+1=65536$
  - Compute $d = 1/e \bmod \phi(n)$
  - Public key is (n,e) and private key is (n,d)
- Sign
  - Compute $s = H(M)^d \bmod n$. The signature is s
- Verify
  - Compute $s^e \bmod n$. Accept iff this equals $H(M) \pmod n$

Why it works: $s^e \equiv \left(H(M)^d\right)^e \equiv H(M)^{de} \equiv H(M) \bmod n$

since $de \equiv 1 \bmod \phi(n)$, then $x^{de} \equiv x \bmod n$, for any x

RSA security

- This simplified version is not secure. Need padding (as in PKCS #1)
  - RSA is deterministic
  - RSA is malleable
    - $Sign(M_1 M_2) \equiv (M_1 M_2)^d \equiv M_1^d M_2^d \equiv Sign(M_1) Sign(M_2)$
  - Several practical attacks exist on early versions of RSA

- Security of RSA relies on hardness of integer factorization
  - Though not proven to be equivalent
  - Best algorithms are versions of general number field sieve

ECDSA

- KeyGen
  - Let E be an elliptic curve with point G of order q
  - Private key is random d, Public key is $Q = [d]G$
- Sign
  - Let k be random mod q. Compute $R=[k]G$
  - Let r be the x-coordinate of R
  - Compute $s = (H(M)+rd)/k \bmod q$
  - The signatures is (r, s)
- Verify
  - Compute $R' = [H(m)/s]G + [r/s]Q$
  - Let r' be the x-coordinate of R'. Accept iff $r' = r$

ECDSA

- KeyGen
  - Let E be an elliptic curve with point G of order q
  - Private key is random d, Public key is $Q = [d]G$
- Sign
  - Let k be random mod q. Compute $R=[k]G$
  - Let r be the x-coordinate of R
  - Compute $s = (H(M)+rd)/k \bmod q$
  - The signatures is (r, s)
- Verify
  - Compute $R' = [H(m)/s]G + [r/s]Q$
  - Let r' be the x-coordinate of R'. Accept iff $r' = r$

Why it works: Exactly the same as DSA, just on curves

Security of ECDSA

- Necessary conditions for the security of DSA
  - The discrete log problem is hard on the elliptic curve
  - *H* is a one-way hash, and is collision-resistant
  - The per-message-secret *k* is unpredictable, and not re-used
- Some validation and other requirements also needed
- Best known attacks on elliptic curve discrete logs are versions of the Pollard Rho or parallel collision search
  - No specialized algorithms that work better than algorithms that work on any group

Appendix 6: The NIST curves

- In FIPS 186-2, NIST recommended 15 elliptic curves of varying security levels, known as “recommended” curves, or more simply, the *NIST curves*
  - *Pseudo-random curves* - coefficients are randomly generated from the output of a seeded cryptographic hash
  - *Special curves* - coefficients and underlying field have been selected to optimize efficiency
- The most widely used curve is P-256, a pseudo-random curve defined over a prime field
  - P-256 provides about 128 bits of classical security
- The NIST curves are also used for key agreement (SP 800-56A)
- We’ll come back to the NIST curves…..

Continued Development

- Before FIPS 186-1, industry implemented RSA signatures following PKCS#1 standard
  - When FIPS 186-1 was developed, NIST assumed industry would switch to ANS X9.31, but this didn't happen
  - NIST moved to allow PKCS#1 version of RSA signatures
- 2009 – FIPS 186-3 increased key sized for DSA and added additional requirements for ECDSA and RSA
  - PKCS #1 version of RSA added
  - NSA collaborated on FIPS 186-3
- 2013 – FIPS 186-4
  - Clarifications and minor corrections
  - Alignment w/ other NIST standards, especially on random number generation

2 Versions of RSA signatures

- In FIPS 186-4, specification for RSA signatures pointed to ANS X9.31 and RFC 8017
  - RFC 8017 also known as PKCS #1 (version 2.1)

- Two RSA signatures defined as in RFC 8017:
  - RSASSA-PKCS1-v1.5
    - Deterministic, no security proof
    - Older, and more widely adopted
  - RSASSA-PSS
    - Randomized, security proof in the ROM
    - More complicated to implement

What else is in FIPS 186-4?

- Besides the signatures, there is a lengthy appendix
  - DSA parameter generation: generation and validation for both provable and probable primes, generation of the generator
  - Criteria for RSA key pairs, generating suitable random provable/probable primes
  - Primality testing
  - Random number generation for the signatures
  - Conversions between data types
  - Various number theoretic algorithms needed to implement the signature schemes
  - Specifications of the NIST curves
  - References
  - Etc….

Curve Concerns

- Efficiency
  - NIST curves chosen to be efficient
  - New curves with more efficient implementations have since been found
- Security
  - The addition operation for the NIST curves has special cases which can allow for side-channel attacks
  - New curves have been found which avoid this pitfall
- Do the NIST curves have hidden weakness?
  - 2013 - Snowden leaks. Reference to dual EC-DRBG?

Types of Curves

- Two different kinds of curves:
  - *Pseudo-random curves* - coefficients are generated from the output of a seeded cryptographic hash
    - Equation for curves: $y^2 = x^3 - 3x + b$ (over prime fields)
    - Equation for curves: $y^2 + xy = x^3 + x^2 + b$ (over binary fields)
  - *Special curves* - coefficients and underlying field have been selected to optimize efficiency
    - Equation for curves: $y^2 + xy = x^3 + ax^2 + 1,\ a \in \{0,1\}.$ (over binary fields)
- Concern expressed over provenance of the parameters of pseudo-random curves
  - Where do NIST curve coefficients come from?

Pseudorandom Curve Generation

- Each pseudo-random curve has a parameter *b*
  - The parameter *b* is the output of a one-way function generated from a seed
    - i.e. Roughly *H*(seed)=*b* (it's more complicated of course)
  - Pseudo-random generation specified in ANSI X9.62 and IEEE P1363
- Given the seed, it is easily verified that *b* was generated by this method
- Ensures the elliptic curve cannot be predetermined

Curve Selection

- In general, a pseudorandom curve was chosen by:
  1) Select a seed and generate the elliptic curve
  2) Check if curve is secure against known attacks. If vulnerable, go to step 1 and repeat

     Note: Very likely need to choose many seeds

- The curves were generated by the NSA
- The seeds and curve parameters are published in the appendix of FIPS 186

- Note: FIPS 186-4 allowed generating your own curve, but this hasn't seen much use in practice

Generation Algorithm

- Appendix C.3.1 in SP 800-186 (Prime case)

Process:

Let l be the bit length of p, and define

$$v = \lfloor (l-1)/hdigest \rfloor$$
$$w = l - hdigest \times v - 1$$

1. Choose an arbitrary *hdigest*-bit string s as the domain parameter *Seed*.
2. Compute $H = HASH(s)$.
3. Let H_0 be the bit string obtained by taking the w rightmost bits of H.
4. Let z be the integer whose binary expansion is given by the *hdigest*-bit string s.
5. For i from 1 to v:
   5.1 Define the *hdigest*-bit string s_i to be the binary expansion of the integer $(z + i) \bmod (2^{hdigest})$.
   5.2 Compute $h_i = HASH(s_i)$.
6. Let h be the bit string obtained by the concatenation of $h_0, h_1, \ldots, h_v$ as follows:

   $$h = h_0 \| h_1 \| \ldots \| h_v.$$

7. Let c be the integer whose binary expansion is given by the bit string h.
8. If $((c = 0)$ or $(4c + 27 \equiv 0 \bmod p))$, then go to Step 1.
9. Choose integers $a, b \in GF(p)$, such that

   $$c\,b^2 \equiv a^3 \bmod p.$$

   (The simplest choice is $a = c$ and $b = c$. However, they may be chosen differently for performance reasons. For example, the pseudorandom prime curves in this document all have $a = -3$.)
10. Check that the elliptic curve E over $GF(p)$ given by $y^2 = x^3 + ax + b$ has suitable order. If not, go to Step 1.

Example: P-256

3.2.1.3. P-256

The elliptic curve P-256 is a Weierstrass curve $W_{a,b}$ defined over the prime field GF(p) that has order $h \cdot n$, where $h = 1$, and n is a prime number. The quadratic twist of this curve has order $h_1 \cdot n_1$, where $h_1 = 3 \times 5 \times 13 \times 179$, and n_1 is a prime number. This curve has domain parameters D = (p, h, n, *Type*, a, b, G, {*Seed*, c}), where the *Type* is "Weierstrass curve," and the other parameters are defined as follows:

p: $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
= 115792089210356248762697446949407573530\
086143415290314195533631308867097853951
(=0xffffffff 00000001 00000000 00000000 00000000 ffffffff ffffffff ffffffff)

h: 1

n: 115792089210356248762697446949407573529\
996955224135760342422259061068512044369
(=0xffffffff 00000000 ffffffff ffffffff bce6faad a7179e84 f3b9cac2 fc632551)

tr: 89188191154553853111372247798585809583
(=$(p+1) - h \cdot n$ = 0x43190553 58e8617b 0c46353d 039cdaaf)

a: −3
= 115792089210356248762697446949407573530\
086143415290314195533631308867097853948
(=0xffffffff 00000001 00000000 00000000 ffffffff ffffffff fffffffc)

b: 41058363725152142129326129780047268409\
114441015993725554835256314039467401291
(=0x5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b)

G_x: 48439561293906451759052585252797914202\
762949526041747995844080717082404635286
(=0x6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296)

G_y: 36134250956749795798585127919587881956\
611106672985015071877198253568414405109
(=0x4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5)

Seed: 0xc49d3608 86e70493 6a6678e1 139d26b7 819f7e90

c: 57436011470200155964173534038266061871\
440426244159038175955947309464595790349
(=0x7efba166 2985be94 03cb055c 75d4f7e0 ce8d84a9 c5114abc af317768 0104fa0d)

Security of NIST curves

- Assuming that SHA-1 cannot be inverted, generation process provides assurance NIST curves not intentionally constructed with hidden weaknesses
- In particular, the NIST curves do NOT belong to any known class of elliptic curves with weak security properties
  - No sufficiently large classes of weak curves are known
- There are NO known attacks of cryptographic significance which lessen the claimed security levels of the NIST curves
- 2015 – NIST workshop on Elliptic Curves Cryptography Standards

FIPS 186-5

- 2015 - NIST opened FIPS 186-4 for public comments
- 2019 – draft FIPS 186-5 published, feedback requested
- 2023 – NIST published FIPS 186-5

What changed in FIPS 186-5

- DSA no longer approved for signature generation
- X9.31 RSA signatures removed
- Larger key sizes for RSA signatures allowed
- The EdDSA signature algorithm is included
- Deterministic version of ECDSA included
- More elliptic curve details added
  - New SP 800-186 has most of them
  - Two new elliptic curves specified
  - Brainpool and secp256k1 curves allowed in an appendix
- Various minor improvements/corrections to algorithms in appendices
  - Allows construction of primes to satisfy restrictions mod 8
  - Allows trial division before checking primality
  - Updated algorithms in Appendices B and C to prevent bias

DSA Signatures

- Two of the domain parameters for DSA are prime numbers *p* and *q*
  - These primes are supposed to be generated deterministically, from a *seed*
- Recent research showed that DSA primes could be generated in such a way that there is a trapdoor
  - With knowledge of the trapdoor, one can compute discrete logs efficiently, which breaks the security of DSA
  - It seems hard to detect if such a trapdoor is present
- Recommended remedy: publish the seed
  - The trapdoor primes have to be specially constructed; publishing the seed shows this wasn't done
- Early version of FIPS 186-5 made publishing the seeds mandatory
- DSA signatures no longer approved for signature generation
  - (Increasing advances in discrete logarithm algorithms)

RSA signatures

- FIPS 186-4 includes RSA signatures using X9.31 and PKCS #1
- ANSI X9.31 was withdrawn, and its are algorithms are now disallowed
  - The padding scheme was shown to be insecure
  - It also included PRNGs -- we have updated guidance in the SP 800-90 series
- FIPS 186-4 required RSA key sizes of length 1024, 2048, or 3072 bits
- FIPS 186-5 to allow any key size with (even) length $\geq$ 2048

Elliptic Curve Crypto

- FIPS 186-4 included an elliptic curve analogue of DSA, called ECDSA
  - Mostly referred to ANSI X9.62 for specific details
  - Included specifications of the NIST curves
- ANSI X9.62 was withdrawn, so for FIPS 186-5 we added back in the details needed to implement ECDSA
- In addition, we are adding new elliptic curve signature algorithms (Deterministic ECDSA and EdDSA) and elliptic curves (Ed25519 and Ed448).
- Many of the elliptic curve details are in a new companion document SP 800-186
  - Including specification of the NIST curves
  - Binary curves now deprecated
  - Brainpool and secp256k1 curves allowed in an appendix

SP 800-186

- Elliptic Curve Parameters for all NIST ECC algorithms
  - Models of curves: Weierstrass, Montgomery, Edwards
  - Curve arithmetic, conversions between models
  - Specifications for NIST-recommended curves
  - Curve generation details
- Implementation aspects
  - Validation and compression

Table 2. Allowed Usage of the Specified Curves

Specified Curves	Allowed Usage
K-233, B-233 K-283, B-283 K-409, B-409 K-571, B-571	Deprecated
P-224 P-256 P-384 P-521	ECDSA, EC key establishment (see [SP_800-56A])
Edwards25519 Edwards448	EdDSA
Curve25519, W-25519 Curve448, E448, W-448	Alternative representations included for implementation flexibility. Not to be used for ECDSA or EdDSA directly.

Table 1. Approximate Security Strength of the Recommended Curves

Security Strength	Recommended Curves
112	P-224, K-233, B-233
128	P-256, W-25519, Curve25519, Edwards25519, K-283, B-283
192	P-384, K-409, B-409
224	W-448, Curve448, Edwards448, E448
256	P-521, K-571, B-571

Deterministic Signatures

- The past decade has seen some attacks which resulted from bad random number generation in signature schemes
- Deterministic signatures desired (for some applications)
  - Deterministic schemes need to be carefully protected against side-channel attacks, particularly in hardware implementations
- **Two deterministic schemes to be added in FIPS 186-5**
  1. Deterministic ECDSA: Following IETF RFC 6979, instead of generating the per-message-secret *k* randomly, generate it deterministically, and follow the rest of ECDSA unchanged.
  2. EdDSA (see the next slides)

Edwards Curves

- The NIST curves are all in Weierstrass form
- For example, the prime curves look like:

$$y^2 = x^3 - 3x + b$$

- Recent research in ECC found a new model: Edwards curves

$$x^2 + y^2 = 1 + dx^2y^2$$

- Edwards curves can be implemented faster, and in a uniform way providing easier constant time implementations

EdDSA

- IETF RFC 8032 specified an Edwards curve digital signature algorithm, known as EdDSA.
  - Based off of Schnorr signatures
  - Introduced in 2011 by Bernstein, Duif, Lange, Schwabe,Yang
- 2 sets of parameters:
  - Ed25519, providing approximately 128 bits of security
  - Ed448, which provides approximately 224 bits of security

- EdDSA is deterministic – care must be taken against side channel attacks
- Also includes a “pre-hash” version, which signs *Hash*(*M*), not *M*

- Advantages: speed (2x faster), security, small keys/sigs, deterministic, constant-time
- Disadvantages: non-standard curve, composite order

Schoolbook EdDSA

- Parameters:
  - An Edwards curve: $x^2 + y^2 = 1 + dx^2y^2$, with basepoint *B* of order *n*, over a finite field F_q
- KeyGen
  - Private key is a random string *k*
  - Let *s* = first half of *H(k)*. Public Key is *A*=[*s*]*B*
- Sign
  - Compute *r* = *H(*2nd half of *H(k),M))*. Then let *R*=[*r*]*B*
  - Compute *S* = *r* + *H(R,A,M)s* mod *n*. The signature is *(R,S)*
- Verify
  - Check if [*S*]*B* = *R*+[*H(R,A,M)*]*A*
  - Works as [*S*]*B* = [*r*+*H(R,A,M)s*]*B* = [*r*]*B*+[*H(R,A,M)s*]*B* = *R*+[*H(R,A,M)*]*A*

Security of EdDSA

- Like ECDSA, relies on the hardness of the elliptic curve discrete log problem
  - Per-message-secret is deterministic from the private key and message
  - Best known attacks same as for ECDSA
- Unlike NIST curves, group order is not prime (small cofactor)
- Easier to implement in constant time

Comparison of FIPS 186-5 signatures

Security Strength	DSA key size	RSA key size	ECC key size
80	1024	1024	160
112	2048	2048	224
128	3072	3072	256
192	7680	7680	384
256	15360	15360	512

- RSA has slow KeyGen, but fast verification
- ECDSA is fast, EdDSA is faster
- RSA is most widely implemented and supported
  - EdDSA gaining popularity

New Signatures (not in FIPS 186-5)

- All the signatures in FIPS 186-5 are quantum vulnerable
- New Post-Quantum Cryptography signatures
  - SP 800-208: Stateful Hash-based Signatures (XMSS, LMS)
  - FIPS 204: Lattice-based ML-DSA (Dilithium)
  - FIPS 205: Hash-based SLH-DSA (Sphincs+)
  - FIPS 206: Lattice-based FN-DSA (Falcon)
- Onramp – Additional PQC signatures under evaluation
  - 14 candidates being evaluated in 2nd Round