2021-05-27 theory

May 2021

Review on the importance of Elliptic Curves:
They are important since the Discrete Log Problem (find n by knowing P and G, where n is like the secret key and P is like the public key) is hard on elliptic curves.

$$"n" \cdot G = P \longrightarrow \text{Hard to compute } n \text{ from } P \text{ and } G.$$

Nowadays, the best algorithms are very slow trying to do this computation but the "hard" assumption is not mathematically proved yet.
Instead, if you know n and G, even if n is an huge number, you can compute $n \cdot G$ in a fast way (by means of a variant of the Square & Multiply algorithm).

Remainder: $"n" \cdot G$ means that the generator G is added n times $(G+G+...+G)$ where '+' is the calculation sign used to add points in EC (Elliptic Curves).

Review on Digital Signatures:
A digital signature is a cryptographic primitive acting as a digital counterpart of an handwritten signature.
Properties:

- Non-repudiation
- Authentication: a signature is an algorithm that needs a public key to perform the verification and so it is necessary also to have a certificate to demonstrate that we are the owner of the public key.
- Integrity
- Unforgeability: if attacker knows some pairs (message, sign) and still cannot build a new valid signature

An example of use of digital signatures is signing a pdf document by means of the software Acrobat PRO.

12.4 EdDSA

Extract from EdDSA original paper:

EdDSA parameters. EdDSA has seven parameters: an integer $b \geq 10$; a cryptographic hash function H producing $2b$-bit output; a prime power q congruent to 1 modulo 4; a $(b-1)$-bit encoding of elements of the finite field $\mathbf{F}_q$; a non-square element d of $\mathbf{F}_q$; a prime ℓ between 2^{b-4} and 2^{b-3} satisfying an extra constraint described below; and an element $B \neq (0,1)$ of the set

$$E = \left\{(x,y) \in \mathbf{F}_q \times \mathbf{F}_q : -x^2 + y^2 = 1 + dx^2y^2\right\}.$$

The condition that d is not a square implies that $d \notin \{0,-1\}$, so this set E forms a group with neutral element $0 = (0,1)$ under the twisted Edwards addition law

$$(x_1,y_1) + (x_2,y_2) = \left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 + x_1x_2}{1 - dx_1x_2y_1y_2}\right)$$

introduced by Bernstein, Birkner, Joye, Lange, and Peters in [13]. Completeness of the addition law — the fact that the denominators $1 \pm dx_1x_2y_1y_2$ are nonzero — follows as explained in [13, Section 6]: -1 is a square in $\mathbf{F}_q$ (since q is congruent to 1 modulo 4), so this addition law on E is $\mathbf{F}_q$-isomorphic to the Edwards addition law on the Edwards curve $x^2 + y^2 = 1 - dx^2y^2$, which is complete by [14, Theorem 3.3] since $-d$ is not a square in $\mathbf{F}_q$. The latter follows from d being a non-square and -1 being a square in $\mathbf{F}_q$. The extra constraint mentioned above is that $\ell B = 0$, where nB means the nth multiple of B in this group.

We use the encoding of $\mathbf{F}_q$ to define some field elements as being negative: specifically, x is negative if the $(b-1)$-bit encoding of x is lexicographically larger than the $(b-1)$-bit encoding of $-x$. If q is an odd prime and the encoding is the little-endian representation of $\{0,1,\ldots,q-1\}$ then the negative elements of $\mathbf{F}_q$ are $\{1,3,5,\ldots,q-2\}$.

An element $(x,y) \in E$ is encoded as a b-bit string $\underline{(x,y)}$, namely the $(b-1)$-bit encoding of y followed by a sign bit; the sign bit is 1 iff x is negative. This encoding immediately determines y, and it determines x via the equation $x = \pm\sqrt{(y^2-1)/(dy^2+1)}$.

EdDSA keys and signatures. An EdDSA secret key is a b-bit string k. The hash $H(k) = (h_0,h_1,\ldots,h_{2b-1})$ determines an integer

$$a = 2^{b-2} + \sum_{3 \leq i \leq b-3} 2^i h_i \in \left\{2^{b-2}, 2^{b-2}+8, \ldots, 2^{b-1}-8\right\},$$

which in turn determines the multiple $A = aB$. The corresponding EdDSA public key is $\underline{A}$. Bits $h_b,\ldots,h_{2b-1}$ of the hash are used as part of signing, as discussed in a moment.

The signature of a message M under this secret key k is defined as follows. Define $r = H(h_b,\ldots,h_{2b-1},M) \in \left\{0,1,\ldots,2^{2b}-1\right\}$; here we interpret $2b$-bit strings in little-endian form as integers in $\left\{0,1,\ldots,2^{2b}-1\right\}$. Define $R = rB$. Define $S = (r + H(\underline{R},\underline{A},M)a) \bmod \ell$. The signature of M under k is then the $2b$-bit string $(\underline{R},\underline{S})$, where $\underline{S}$ is the b-bit little-endian encoding of S. Applications wishing to pack data into every last nook and cranny should note that the last three bits of signatures are always 0 because ℓ fits into $b-3$ bits.

Verification of an alleged signature on a message M under a public key works as follows. The verifier parses the key as $\underline{A}$ for some $A \in E$, and parses the alleged signature as $(\underline{R},\underline{S})$ for some $R \in E$ and $S \in \{0,1,\ldots,\ell-1\}$. The verifier computes $H(\underline{R},\underline{A},M)$ and then checks the group equation $8SB = 8R + 8H(\underline{R},\underline{A},M)A$ in E. The verifier rejects the alleged signature if the parsing fails or if the group equation does not hold.

To see that signatures pass verification, simply multiply B by the equation $S = (r + H(\underline{R},\underline{A},M)a) \bmod \ell$, and use the fact that $\ell B = 0$, to see that $SB = rB + H(\underline{R},\underline{A},M)aB = R + H(\underline{R},\underline{A},M)A$. The verifier is *permitted* to check this stronger equation and to reject alleged signatures where the stronger equation does not hold. However, this is not *required*; checking that $8SB = 8R + 8H(\underline{R},\underline{A},M)A$ is enough for security.

Notes:

- B is going to be the generator (with order l) of the elliptic curve.
- The E curve is not in its Weierstrass form above. Its Weierstrass form is $y^2x^3 + ax + b(modp)$ and it's equivalent to the one shown in the paper.
- In the Edwards curve, if $d = r^2$, then when this curve is transformed into its Weierstrass form, it will not respect the $\neq 0$ statement. For example: $4 \cdot a^3 + 27b^2 \neq 0(modp)$ would not be respected. So, d should be always different from r^2.
- The equation $8SB8R + 8H(R, A, M)A$ is equivalent in its form without the 8 constants. This is due to the fact that the order l is prime and so an equation like $8X8Y$ is equivalent to $X = Y$.

EdDSA signatures are deterministic (if we sign twice the same message/document with the same key, we will obtain the same signature). This protects against attacks arising from signing with bad randomness; the effects of which can, depending on the algorithm, range up to full private key compromise.

12.5 BLS Signature

A BLS digital signature — also known as Boneh–Lynn–Shacham (BLS) — is cryptographic signature scheme which allows a user to verify that a signer is authentic.
The scheme uses a bilinear pairing for verification, and signatures are elements of an elliptic curve group. Working in an elliptic curve group provides some defense against index calculus attacks, allowing shorter signatures than FDH (Full Domain Hash) signatures for a similar level of security.
Signatures produced by the BLS signature scheme are often referred to as short signatures, BLS short signatures, or simply BLS signatures. The signature scheme is provably secure (the scheme is existentially unforgeable under adaptive chosen-message attacks).
Short signature scheme is designed for systems where signatures are typed in by a human or signatures are sent over a low-bandwidth channel.

A signature scheme consists of three functions: generate, sign, and verify.
Key generation:
The key generation algorithm selects a random integer x such as $0 < x < r$. The private key is x. The holder of the private key publishes the public key, g^x.
Signing:
Given the private key x, and some message m, we compute the signature by hashing the bitstring m, as $h = H(m)$. We output the signature $\sigma = h^x$.
Verification:
Given a signature σ and a public key g^x, we verify that $e(\sigma, g) = e(H(m), g^x)$.

12.6 EC attacks

12.6.1 Implementation

NOTE 12.6.1

Invalid curve attacks. This attack takes advantage that users do not verify that points belong to the safe elliptical curve. For example, a server receives a point P and calculates $s \cdot P$ (where s is the server's secret key) without verifying that P belongs to the elliptical curve in the protocol. It could happen that the point P belongs to a curve with few points from where it would be possible to calculate a remainder of s modulo the order of P. By repeating this type of request, with different order points, it may be possible to derive s via the Chinese Remainder Theorem (CRT).

https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

It is crucial to select a different ephemeral key K for different signatures. That is to say, an attacker who knows two signatures (r, s),(r, s') employing the same K can efficiently compute the private key sk.

In December 2010, a group calling itself *fail0verflow* announced recovery of the ECDSA private key used by Sony to sign software for the PlayStation 3 game console. However, this attack only worked because Sony did not properly implement the algorithm, because k was static instead of random. As pointed out in the Signature generation algorithm section above, this makes d_A solvable and the entire algorithm useless.[7]

12.6.2 Mathematical's

MOV attack

Most cryptosystems based on elliptic curves can be broken if you can solve the discrete logarithm problem, that is, given the point P and rP, find the integer r.

The MOV attack uses a bilinear pairing, which (roughly speaking) is a function e that maps two points in an elliptic curve $E(F_q)$ to a element in the finite field F_{q^k}, where k is the embedding degree associated with the curve. The bilinearity means that $e(rP, sQ) = e(P, Q)rs$ for points P, Q. Therefore, if you want to compute the discrete logarithm of rP, you can instead compute $u = e(P, Q)$ and $v = e(rP, Q)$ for any Q. Due to bilinearity, we have that $v = e(P, Q)r = ur$. Now you can solve the discrete logarithm in F_{q^k} (given ur and u, find r) in order to solve the discrete logarithm in the elliptic curve!

Usually, the embedding degree k is very large (the same size as q), therefore transfering the discrete logarithm to F_{q^k} won't help you. But for some curves

the embedding degree is small enough (specially supersingular curves, where $k \leq 6$), and this enables the MOV attack. For example, a curve with a 256-bit q usually offers 128 bits of security (i.e. can be attacked using 2128 steps); but if it has an embedding degree 2, then we can map the discrete logarithm to the field F_{q^2} which offers only 60 bits of security.

In practice the attack can be simply avoided by not using curves with small embedding degree; standardized curves are safe. Since pairings also have many constructive applications, it is possible to carefully choose curves where the cost of attacking the elliptic curve itself or the mapped finite field is the same.

In the end, for SuperSingular curves the DLP (Discrete Logartihm Problem) is equivalent to DLP on $GF(q^k)$ with $k \leq 6$.

NOTE 12.6.2

As explained in https://safecurves.cr.yp.to/twist.html: **a small-subgroup attack** on DH works as follows. Instead of sending a legitimate curve point $e \cdot P$ to Bob, Eve sends Bob a point Q of small order, pretending that Q is her public key. Bob computes $n \cdot Q$ as usual, where n is Bob's secret key; computes a hash of $n \cdot Q$ as a shared secret key for, e.g., AES-GCM; and uses AES-GCM to encrypt and authenticate data. Because Q has small order, there are not many possibilities for $n \cdot Q$; Eve can simply enumerate the possibilities and check which possibility successfully verifies the data. This attack reveals n modulo the order of Q. So as Chinese mathematician Sunzi (3rd century AD) or the Indian mathematician Aryabhata (4th century AD) showed knowledge of n modulo the order of several Q may reveal n.