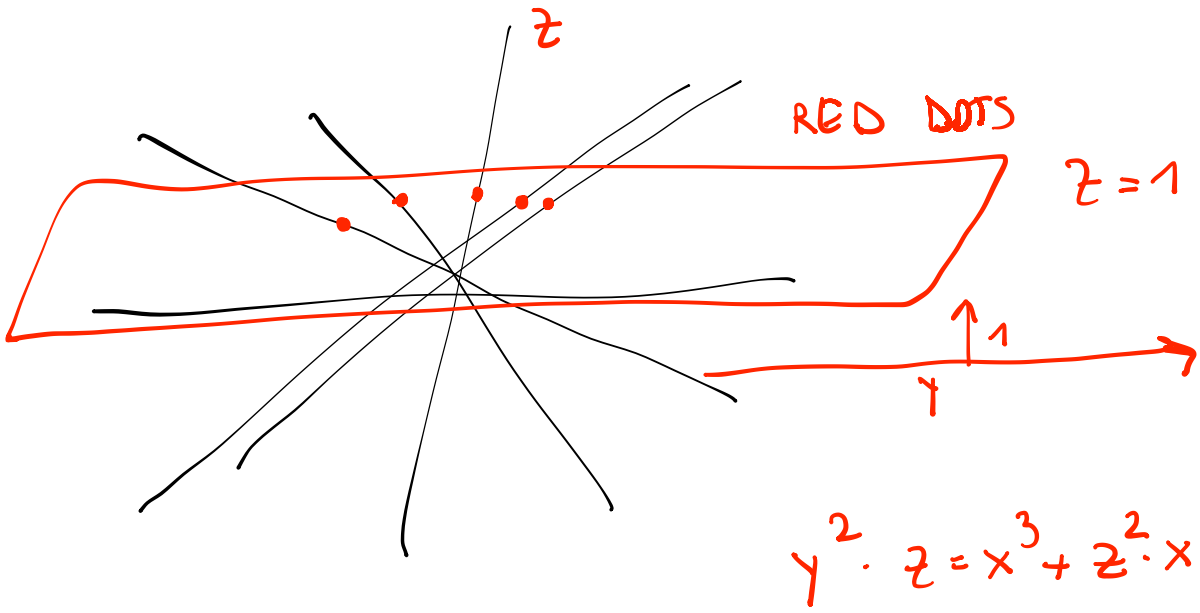
Crypto 12

Note: this material is not intended to replace the live lecture for students.

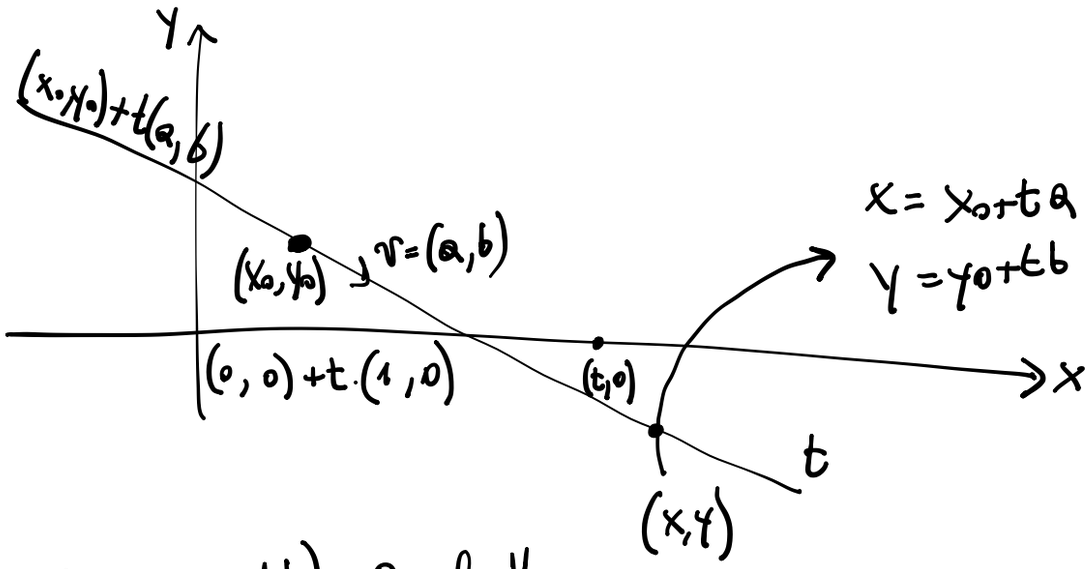
Contents

$$\mathrm{T} = (p\,,\ a\,,\ b\,,\ G\,,\ n\,,\ h)$$



cubic →

↓



$P(x_0+ta, y_0+tb) = 0$ hodl

for the value of "t"

$y^2 = x^3 + x \rightarrow P(x,y) = x^3 + x - y^2$

$x = x_0 + ta$

$y = y_0 + tb$

$0 = (x_0+ta)^3 + (x_0+ta) - (y_0+tb)^2$

Solution for "t" $t^3 \cdot a^3 + \underbrace{\quad\quad}_{t^2 \quad t}$

12.1 EC-DH

12.1.1 Elliptic curve Diffie-Hellman (ECDH)

Alice and **Bob** agree to use an elliptic curve E with parameters domain $\mathrm{T} = (p\,,\,a\,,\,b\,,\,G\,,\,n\,,\,h)$.

Alice		**Bob**
Pick secret key $\mathsf{A} \in \{2, \cdots, \#\mathrm{E}-1\}$		
Compute public $\mathsf{pA} = \mathsf{A} \cdot G = (x_A, y_A)$		
	$\xrightarrow{\mathsf{pA}}$	
		Pick secret key $\mathsf{B} \in \{2, \cdots, \#\mathrm{E}-1\}$
		Compute public $\mathsf{pB} = \mathsf{B} \cdot G = (x_B, y_B)$
	$\xleftarrow{\mathsf{pB}}$	
$\mathsf{k}_A = \mathsf{A} \cdot \mathsf{pB}$		$\mathsf{k}_B = \mathsf{B} \cdot \mathsf{pA}$

So $\mathsf{k} = \mathsf{k}_A = \mathsf{k}_B$ is the session key between **Alice** and **Bob**.

A and B are the private keys whilst pA, pB are the public keys.

12.1.2 Example ECDH

Alice and **Bob** agree to use an elliptic curve $\mathrm{E} : y^2 = x^3 + 2x + 2 \pmod{17}$ and use as generator $G = (5, 1)$. So the parameters domain T are $\mathrm{T} = (17\,,\,2\,,\,2\,,\,(5,1)\,,\,19\,,\,1)$.

Alice		**Bob**
Pick secret key $\mathsf{A} = 7$		
Compute public $\mathsf{pA} = 7 \cdot G = (0, 6)$		
	$\xrightarrow{\mathsf{pA}=(0,6)}$	
		Pick secret key $\mathsf{B} = 13$
		Compute public $\mathsf{pB} = 13 \cdot G = (16, 4)$
	$\xleftarrow{\mathsf{pB}=(16,4)}$	
$\mathsf{k}_A = \mathsf{A} \cdot \mathsf{pB} = 7 \cdot (16, 4)$		$\mathsf{k}_B = 13 \cdot (0, 6)$

So the session key between **Alice** and **Bob** is $\mathsf{k} = (3, 16)$

12.1.3 multiplication $n \cdot P$ on E

```
''' mult(n,P,a,b,q) = (x1,y1) .
        where (x1,y1) is the addition of P n-times on
        the elliptic curve y^2 = x^3 + ax + b  (mod q)
'''
def mult(n,P,a,b,q):
    from checkEll import checkEll
    from addition import addition
    if P=="O": return "O"
    if checkEll(P,a,b,q)==False: return "(x,y) not in Ell"
    else:
        T="O"
        if n>0:
            bina='{0:b}'.format(n)
            for d in bina:
                T=addition(T,T,a,b,q)
                if d=='1': T=addition(T,P,a,b,q)
            if checkEll(T,a,b,q)==True: return T
            else: return "error multiplication"
        elif n==0: return "O"
        else:
            H = (x,(-y)%q)
            return mult(-n,H,a,b,q)
```

Exercise 12.1.4

Check that the $7 \cdot (16, 4) = 13 \cdot (0, 6) = (3, 16)$ on the elliptic curve of the previous example 12.1.2.

FF. A

$g \in \mathbb{F}_p$ p is a prime number

$\mathbb{F}_p^* \approx$ Elliptic

$\nearrow_G$ $\lambda = 128$

$p - 1 \geq 3072$

$p - 1 = 2^{3072 \cdot m}$

$\{ g, g^2, \dots, g^n = 1 \} = \langle g \rangle \subset \mathbb{F}_p^*$

$PK \in \langle g \rangle$

$SK \in \mathbb{Z}_m$

(SK, PK) Key - Generator

$g^{SK} = PK$ in $\mathbb{F}_p^* \iff g = PK \bmod p$

The math problem is the discrete logarithm

Elliptic Curves

$G \in \{(x,y) : y^2 = x^3 + ax + b\}$!! $x, y \in \mathbb{F}_p$

$SK \in \mathbb{Z}_n \longrightarrow \langle G \rangle = \{G, \underbrace{G+G}_{2G}, \underbrace{G+G+G}_{3G}, \dots, \underbrace{n \cdot G}_{0}\}$ $n \sim p$

(SK, PK) Key generation $PK \in \langle G \rangle$

$SK \cdot G = PK$ ↔ equal on the elliptic curve

$SK = (d_L \dots d_2 d_1 d_0)_2$ $SK = d_L 2^L + \dots + d_2 2^2 + d_1 2 + d_0$

$G, 2G, 2^2G, 2^3G, \dots, 2^L G$ $L = \log_2 n$

$SK \cdot G$

Exercise 12.1.5

Alice and **Bob** are using the ECDH protocol with the elliptic curve

$$\mathrm{E} : y^2 = x^3 + x + 6 \pmod{11}$$

Alice's secret key is $\mathsf{A} = 6$. She receives **Bob**'s public key $\mathsf{pB} = (5, 9)$.

1) Check that pB is a point of E.

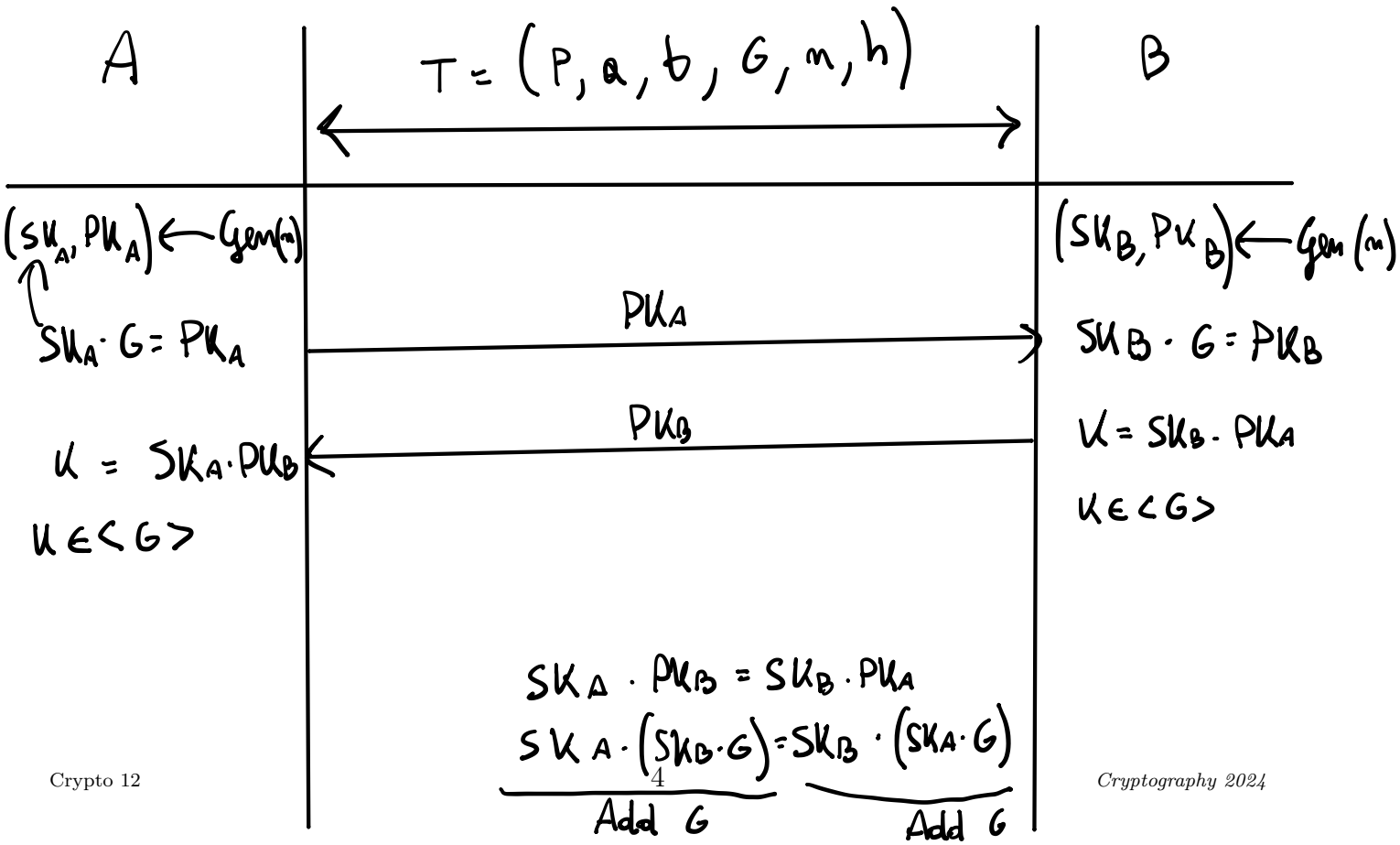
2) Compute the session key k.

Exercise 12.1.6

After using the ECDH protocol **Alice** and **Bob** posses a session key $\mathsf{k} = (x, y)$. The modulus q of the elliptic curve is a 64-bit prime number.
They want to derive a key $\mathsf{k}_{\mathrm{AES}}$ of AES256 by using the hash function SHA256 as follows:

$$\mathsf{k}_{\mathrm{AES}} = \mathrm{SHA256}(x||y)$$

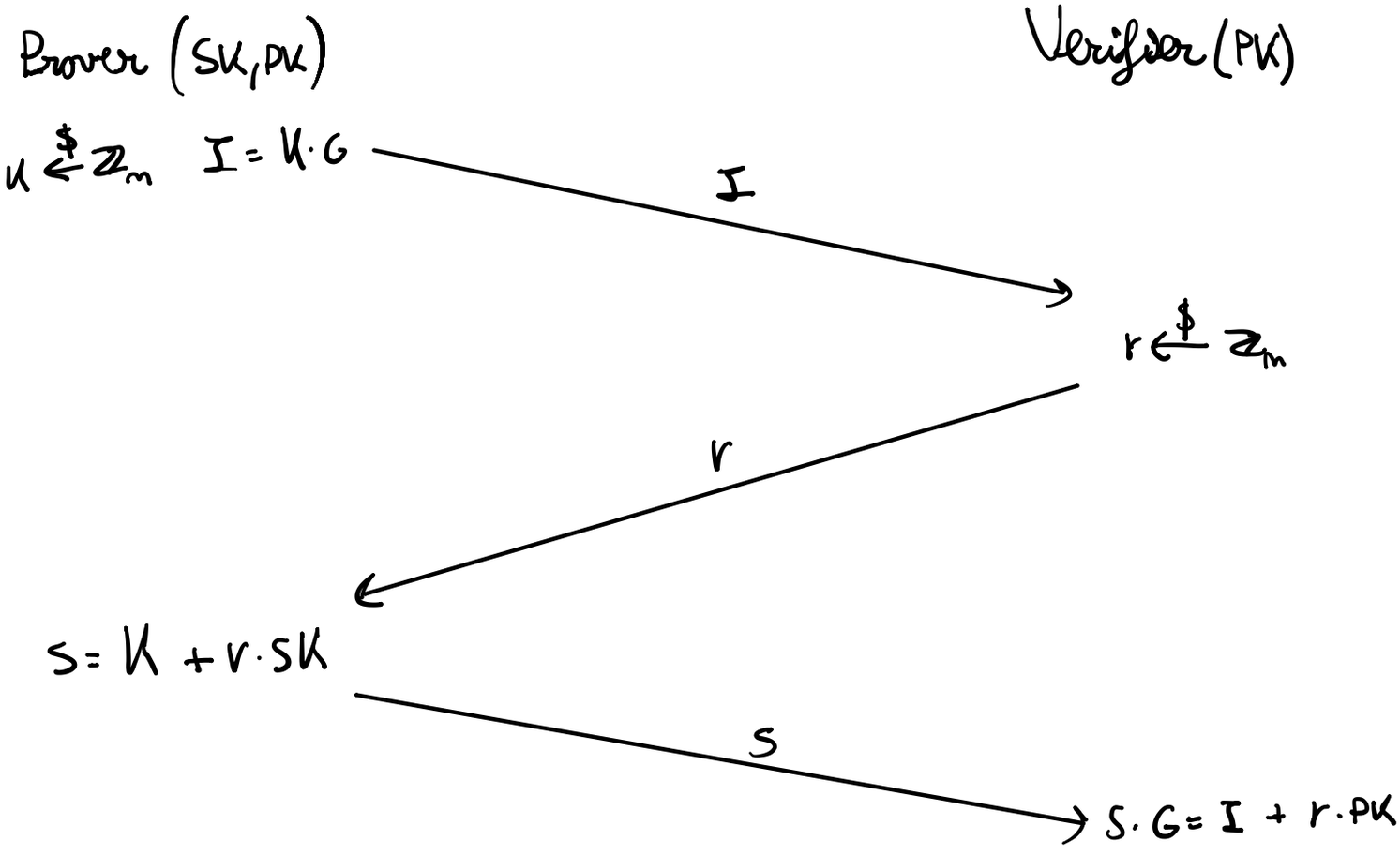
Is this a good idea?



$(SK_A \cdot SK_B)$ $(SK_B \cdot SK_A)$

Schnorr Identification

$T = (p, a, b, G, n, h)$ $X \cdot G = PK$



Verifier accepts if s solve

$I = u \cdot G$

$PK = SK \cdot G$

$X \cdot G = u \cdot G + r \cdot SK \cdot G$

$X \cdot G = (u + r \cdot SK) \cdot G$

Schnorr Digital Signature

Hash Field:

$(SK, PK) \leftarrow Gen(n)$

$\rightarrow Sign_{sk}(M) = \sigma = (r, s)$

$k \xleftarrow{\$} \mathbb{Z}_n$

$I = k \cdot G$

$r = Hash(I \| M)$

$s = k + r \cdot sk$

$\rightarrow Verify((M, \sigma), PK)$

$\sigma = (s, r)$

$s \cdot G - r \cdot PK = I$

$Hash(I \| M) \stackrel{?}{=} r$ → Accept / Reject

```
# Input

aE = 0    # aE integer decimal
bE = 7     # bE  integer decimal


G=(23,23)
orG = 138  # orG integer decimal
q=137
p='{:x}'.format(q)

P=(124,106)
```

$y^2 = x^3 + a_E x + b_E \bmod q$

$sk \cdot G = P$

12.2 EC-Massey-Omura Exchange

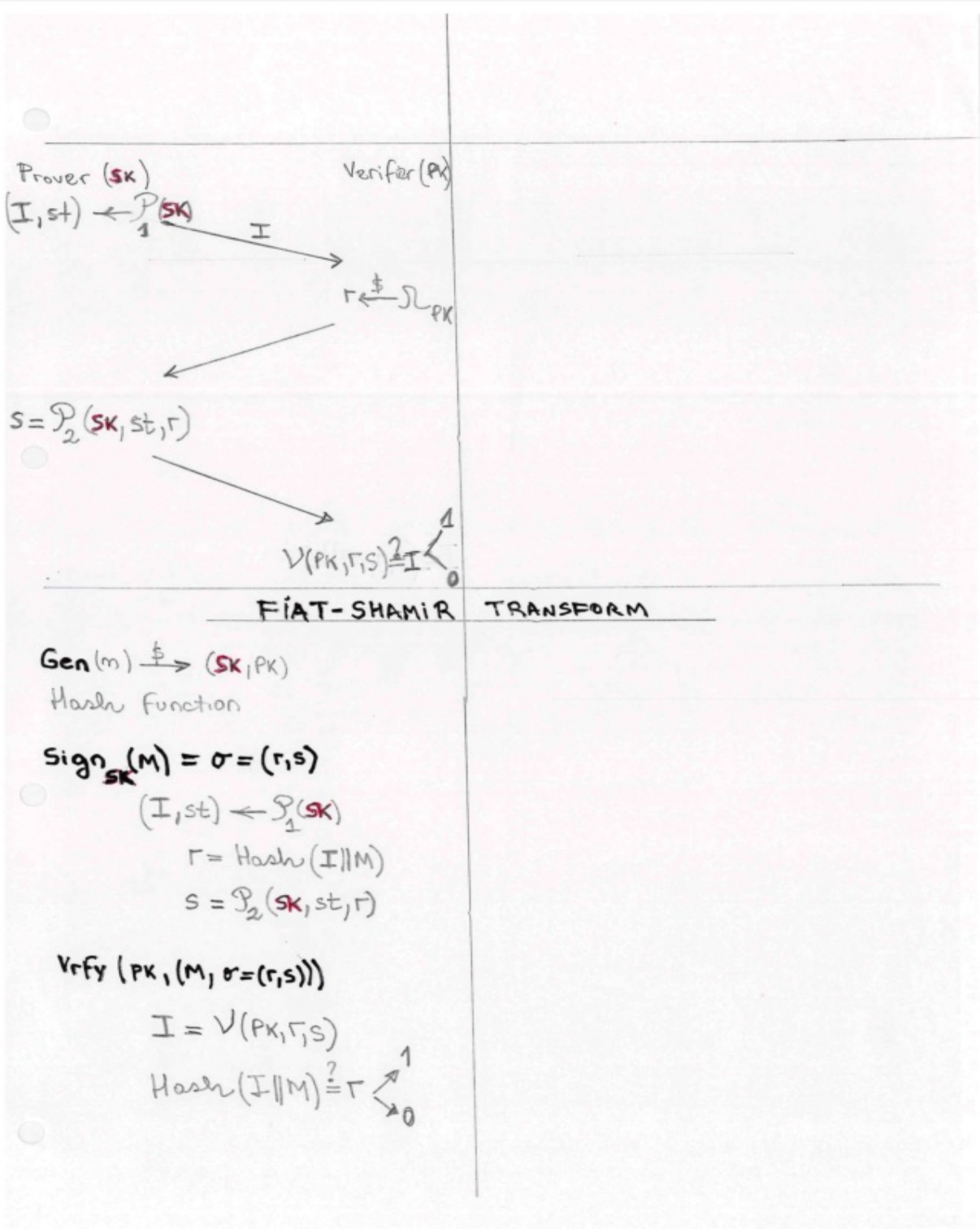
Messages M to be exchanged are encoded as points of an elliptic curve E with n points.

12.2.1 EC-Massey-Omura

Alice has a secret key $0 < \mathsf{A} < n$ such that $\gcd(\mathsf{A}, n) = 1$. So she can efficiently compute $\mathsf{A}^{-1} \pmod{n}$.
Bob has a secret key $0 < \mathsf{B} < n$ such that $\gcd(\mathsf{B}, n) = 1$. So he can efficiently compute $\mathsf{B}^{-1} \pmod{n}$.

Alice		**Bob**
Compute $\mathrm{C}_0 = \mathsf{A} \cdot M$		
	$\xrightarrow{\mathrm{C}_0}$	
		Compute $\mathrm{C}_1 = \mathsf{B} \cdot \mathrm{C}_0$
	$\xleftarrow{\mathrm{C}_1}$	
Compute $\mathrm{C}_2 = \mathsf{A}^{-1} \cdot \mathrm{C}_1$		
	$\xrightarrow{\mathrm{C}_2}$	
		Compute $M = \mathsf{B}^{-1} \cdot \mathrm{C}_2$

12.3 EC-Schnorr digital signature



12.4 EC-DSA

https://goteleport.com/blog/comparing-ssh-keys

ECDSA & EdDSA

The two examples above are not entirely sincere. Both Sony and the Bitcoin protocol employ ECDSA, not DSA proper. ECDSA is an elliptic curve implementation of DSA. Functionally, where RSA and DSA require key lengths of 3072 bits to provide 128 bits of security, ECDSA can accomplish the same with only 256-bit keys. However, ECDSA relies on the same level of randomness as DSA, so the only gain is speed and length, not security.

The ECDSA protocol is quite similar to the standard DSA. So assume **Alice** wants to sign a message M. She pick an elliptic curve e.g. $\mathrm{T} = (p\,,\, a\,,\, b\,,\, G\,,\, n\,,\, h)$, where the order n of G is a prime number. Here it is how she generate the keys:

12.4.1 Key generation for ECDSA

1) Choose a RND integer $d \in \{1, 2, \cdots, n-1\}$.
2) Compute $B = d \cdot G$.

The keys are:
$\mathsf{pk} = (p, a, b, n, G, B)$;
$\mathsf{sk} = d$

Now **Alice** signs the message M:

12.4.2 ECDSA Signature Generation

1) Choose an RND integer K as ephemeral key with $0 < \mathsf{K} < n$.
2) Compute $R = \mathsf{K} \cdot G = (x_R, y_R)$ and set $r = x_R$.
3) Compute $s = (\mathsf{hash}(M) + d \cdot r) \cdot \mathsf{K}^{-1} \pmod n$. If $s = 0 \pmod n$ go to 1).

The signature $\mathsf{Sign}_{\mathsf{sk}}(M)$ of M is the pair (r, s).

Here the verification process:

12.4.3 ECDSA Verification

1) Compute the auxiliar $w = s^{-1} \pmod n$.
2) Compute the auxiliar $u_1 = w \cdot \mathsf{hash}(M) \pmod n$.
3) Compute the auxiliar $u_2 = w \cdot r \pmod n$.
4) Compute $P = u_1 \cdot G + u_2 \cdot B = (x_P, y_P)$.

5) The $\mathsf{Vrfy}_{\mathsf{pk}}(M, (r, s))$ outputs 1 for a valid signature if $x_P = r \pmod n$ otherwise $\mathsf{Vrfy}_{\mathsf{pk}}(M, (r, s))$ outputs 0.

To proof that the above verification do work notice that:

$$P = (x_P, y_P) = u_1 \cdot G + u_2 \cdot B = w \cdot \mathsf{hash}(M) \cdot G + w \cdot r \cdot d \cdot G$$

then

$$P = (\mathsf{hash}(M) + r \cdot d) \cdot w \cdot G$$

so if (r, s) are valid we get

$$P = \mathsf{K} \cdot G = (x_R, y_R)$$

hence $x_p = r \pmod n$.

Exercise 12.4.4

Consider ECDSA on the elliptic curve $E(\mathbb{Z}_{17}) : y^2 = x^3 + 2x + 2$ with $G = (5, 1)$, $n = 19$ and $\mathsf{sk} = d = 5$.

Let M be a message with $\mathsf{hash}(M) = 8$.
Sign M and compute the public key $\mathsf{pk} = (p, a, b, n, G, B)$.

Exercise 12.4.5

Explain an identification sigma protocol so that the EC-DSA signature can be regarded as a kind-of Fiat-Shamir transform.

12.5 EC attacks

The security of deployed asymmetric cryptographic schemes relies on the believed hardness of number theoretic problems such as integer factorization and the computation of discrete logarithms in finite fields or in groups of points on an elliptic curve. However, most real-world cryptographic vulnerabilities do not stem from a weakness in the underlying hardness assumption, but rather from implementation issues such as side-channel attacks, software bugs or design flaws.

[Bos13, Introduction]

12.5.1 Implementation

NOTE 12.5.1

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]

12.5.2 Mathematical's

MOV attack : https://crypto.stackexchange.com/questions/1871/how-does-the-mov-attack-work
SuperSingular curves: for such curves the DLP is equivalent to DLP on GF(q^k) with $k \leq 6$ (MOV

attack).

NOTE 12.5.2

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.

12.6 Appendix:

12.6.1 EdDSA

EdDSA Original paper

RFC8032 Standard Implementation of EdDSA

EdDSA

12.6.2 BLS Signature

BLS original paper

Wikipedia: BonehLynnShacham

12.7 Bibliography

Books I used to prepare this note:

[Aumasson18] Jean-Philippe Aumasson, *Serious Cryptography: A Practical Introduction to Modern Encryption*, No Starch Press, 2018.

[Paar10] Paar, Christof, Pelzl, Jan, *Understanding Cryptography, A Textbook for Students and Practitioners*, Springer-Verlag, 2010.

Here a list of papers:

[Bos13] Bos J. W. et al: *Elliptic Curve Cryptography in Practice*, https://eprint.iacr.org/2013/734.pdf

[DH76] Diffie, W.; Hellman, M. *New directions in cryptography*, (1976). IEEE Transactions on Information Theory. 22 (6): 644654.

and some interesting links:

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

https://safecurves.cr.yp.to/