Modern Cryptography: Lecture 10

The Public Key Revolution II/II

Daniel Slamanig

AUSTRIAN INSTITUTE
OF TECHNOLOGY

Organizational

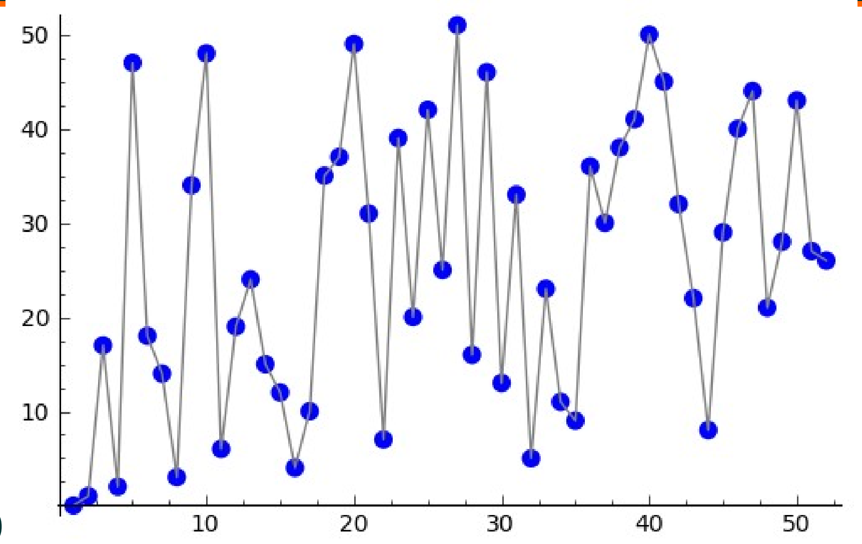
- Where to find the slides and homework?
  - https://danielslamanig.info/ModernCrypto18.html
- How to contact me?
  - daniel.slamanig@ait.ac.at
- Tutor: Karen Klein
  - karen.klein@ist.ac.at

- Official page at TU, Location etc.
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=8632&dsrid=679&courseNr=192062&semester=2018W
- Tutorial, TU site
  - https://tiss.tuwien.ac.at/course/courseAnnouncement.xhtml?dswid=5209&dsrid=341&courseNumber=192063&courseSemester=2018W

- Exam for the second part: Thursday 31.01.2019 15:00-17:00 (Tutorial slot)
  - No tutorial this week → exam for first part

Discrete Logarithms

- We consider a cyclic group G of order q with generator g, so G = $\{g^0, \ldots, g^{q-1}\}$
- The DL problem: given $h=g^x$ to find the unique $x \in \mathbb{Z}_q$
- Let $\mathcal{G}$ be a group generator that on input 1n outputs a description of a cyclic group (G, q, g) with ‖q‖=n (binary length)



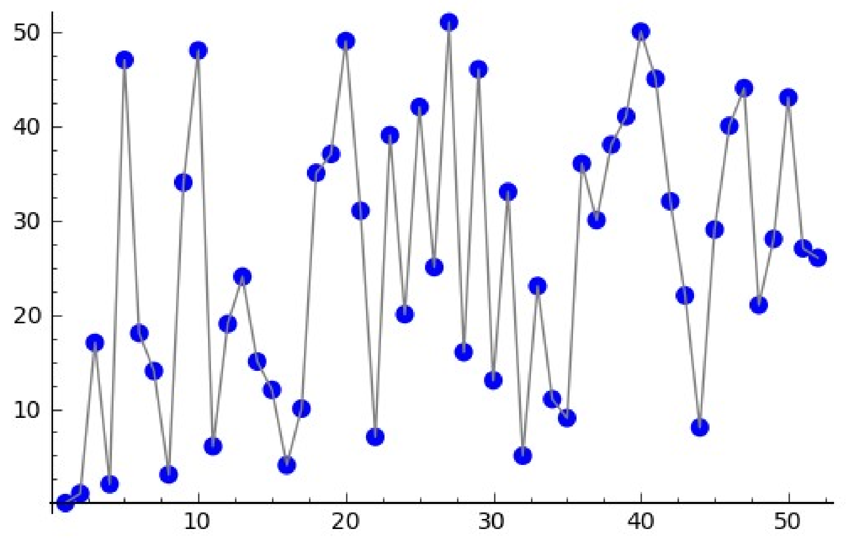
$\mathbb{Z}^*_{53}$, g=2

The discrete-logarithm experiment $\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n)$:

1. Run $\mathcal{G}(1^n)$ to obtain (G, q, g), where G is a cyclic group of order q (with ‖q‖ = n), and g is a generator of G.
2. Choose a uniform $h \in G$.
3. $\mathcal{A}$ is given G, q, g, h, and outputs $x \in \mathbb{Z}_q$.
4. The output of the experiment is defined to be 1 if $g^x = h$, and 0 otherwise.

Discrete Logarithms

- We consider a cyclic group G of order q with generator g, so G = $\{g^0, \ldots, g^{q-1}\}$
- The DL problem: given h=g^x to find the unique x ∈ $\mathbb{Z}_q$
- Let $\mathcal{G}$ be a group generator that on input 1n outputs a description of a cyclic group (G, q, g) with ‖q‖=n (binary length)



$\mathbb{Z}^*_{53}$, g=2

The discrete-logarithm experiment $\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n)$:

1. Run $\mathcal{G}(1^n)$ to obtain (G, q, g), where G is a cyclic group of order q (with ‖q‖ = n), and g is a generator of G.

2. Choose a uniform h ∈ G

DEFINITION 8.62 We say that the discrete-logarithm problem is hard relative to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathrm{negl}(n).$$

Problems Related to the DLOG Problem

- We will now take a look at two problems related but weaker than the DLP; the computational (CDH) and the decisional Diffie–Hellman (DDH) problem
- Let **DH**$_g(h_1, h_2) := g^{\log_g h_1 \cdot \log_g h_2}$
  - If $h_1 = g^{x_1}$ and $h_2 = g^{x_2}$, then **DH**$_g(h_1, h_2) = g^{x_1 x_2} = h_1^{x_2} = h_2^{x_1}$
- CDH Problem
  - Given (G, q, g, h_1, h_2) compute **DH**$_g(h_1, h_2)$

DEFINITION: We say that the CDH problem is hard relative to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ there is a negligible function negl such that

$$\Pr[\mathcal{A}(G, q, g, g^x , g^y) = g^{xy}] \leq \mathrm{negl}(n),$$

where the probabilities are taken over the experiment in which $\mathcal{G}(1^n)$ outputs (G, q, g), and then uniform x, y $\mathbb{Z}_q$ are chosen.

Problems Related to the DLOG Problem

- DDH Problem
  - Given (G, q, g) and uniform random $h_1, h_2 \in G$, distinguish $\mathbf{DH}_g(h_1, h_2)$ from uniformly random $h' \in G$

DEFINITION 8.63: We say that the DDH problem is hard relative to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ there is a negligible function negl such that

$$\Pr[\mathcal{A}(G, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(G, q, g, g^x, g^y, g^{xy}) = 1] \leq \text{negl}(n),$$

where in each case the probabilities are taken over the experiment in which $\mathcal{G}(1^n)$ outputs (G, q, g), and then uniform $x, y, z \in \mathbb{Z}_q$ are chosen.

Clearly, if we can solve DL, then we can solve DDH and CDH

DDH is a stronger assumption than CDH (HW)

There are groups where the CDH is assumed hard, but the DDH is easy (HW)

Algorithms for Computing Discrete Logarithms

- Two types of algorithms
  - Generic ones: apply to arbitrary groups
  - Specific ones: tailored to work for some specifc class of groups

Generic for groups of order q:

-Baby step/giant step (Shanks)*: $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q))$ time and $\mathcal{O}(\sqrt{q})$ space
-Pollard's rho*: $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q))$ time and constant space

Generic for groups of order q (if factorization is known/easy to compute):

-Pohlig-Hellman: Reduces to finding DL in group or order q' with q' the largest prime dividing q (use then any algorithm to solve the DL)

Specific algorithm for $\mathbb{Z}^*_p$:

-Index Calculus/Number Field Sieve: Subexponential with runtime $2^{\mathcal{O}((\log p) \cdot (\log \log p))}$

* time complexity optimal for generic algorithms

The Baby-Step/Giant-Step Algorithm I/II

- Want to solve DL problem for some h=g^x in (G, q, g)
- We know that h must lie somwhere in the cycle {g^0, ..., g^{q-1}}
  - Computing all elements would take Ω(q) time!

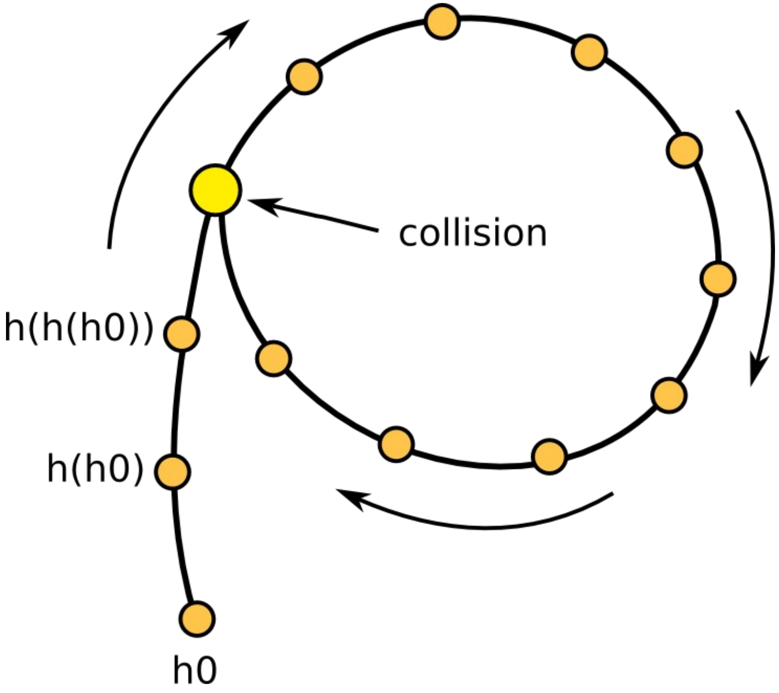
- Take some elements of the cycle at steps t=$\lfloor\sqrt{q}\rfloor$ (the “giant steps”)
  - Gives us a list (g^0 , g^t , g^{2t} , ... , $g^{\lfloor q/t\rfloor\cdot t}$) with gaps of at most t elements
  - We know h lies in one of the gaps
  - Compute a list (h·$g^{1,}$ ..., h·g^t) of shifts of h (the “baby steps”)
  - One of the points in the “baby list” will be equal to one in the “giant list”, i.e., h·g^i = $g^{k\cdot t}$ for some i and k
  - And determine x = (kt – i) mod q

The Baby-Step/Giant-Step Algorithm II/II

- Complexity
  - $\mathcal{O}(\sqrt{q})$ exponentiations/multiplications
  - Sorting the "giant list" takes $\mathcal{O}(\sqrt{q} \cdot \log q)$
  - Binary search for each element from "baby list" in $\mathcal{O}(\log q)$
  - Overall $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q))$ time <u>but need to store</u> $\mathcal{O}(\sqrt{q})$ elements

- Can we do better generically?

The Pollard Rho Algorithm*



- Idea: Let $H_{g,h}: \mathbb{Z}_q \times \mathbb{Z}_q \to G$ be defined by $H_{g,h}(x_1 , x_2) = g^{x_1} \cdot h^{x_2}$
- The birthday bound says we find a collision in $H_{g,h}$ in time $\mathcal{O}(\sqrt{q})$
- Is possible with constant memory (see §5.4.2)
- If $H_{g,h}(x_1 , x_2) = H_{g,h}(x_1' , x_2')$ with $x_1 \neq x_1$ and $x_2 \neq x_2$ then solve $\gamma(x_2 - x_2') = (x_1' - x_1) \bmod q$ for γ

- Some issues not yet considerd
  - Range of hash function must be subset of its domain: Use a standard cryptographic hash function $F: G \to \mathbb{Z}_q \times \mathbb{Z}_q$ to obtain the input for G

* we use the description from the book for consistency

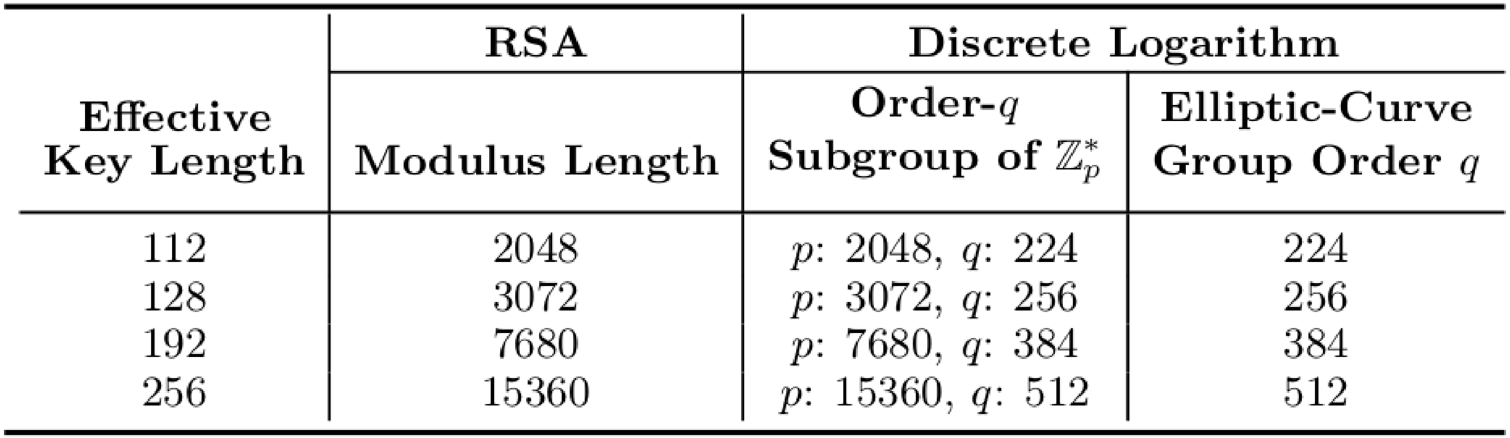
Choice of Discrete Logarithm Hard Groups

- Generic vs. special algorithms
  - If only generic algorithms are available parameters can be chosen much smaller; Yields more efficient group operations
- Prime order vs. composite order groups
  - Prime order: Discrete logarithm problem is hardest in prime order groups and finding generators is trivial
  - Composite order: Need to have subgroup of sufficient size (recall: largest prime dividing the order; may need to consider specific algorithms). Finding generators is more cumbersome.
- Prime order groups are preferable (there are some more reasons why discussed later, see also HW)

Choice of Discrete Logarithm Hard Groups

- Groups that are of interest
  - $\mathbb{Z}^*_p$ (does not have prime order)
  - Prime order q subgroups of $\mathbb{Z}^*_p$
  - Elliptic curve groups

What about $\mathbb{Z}_p$ with addition?



	RSA	Discrete Logarithm	
Effective Key Length	Modulus Length	Order-q Subgroup of $\mathbb{Z}_p^*$	Elliptic-Curve Group Order q
112	2048	p: 2048, q: 224	224
128	3072	p: 3072, q: 256	256
192	7680	p: 7680, q: 384	384
256	15360	p: 15360, q: 512	512

Key sizes recommended by NIST (from §9.3)

Prime Order Subgroups of $\mathbb{Z}^*_p$

- We can “craft” p in a way that it has a prime order q subgroup of desired size

THEOREM 8.64 Let p = rq + 1 with p, q prime. Then

$$G = \{h^r \bmod p \mid h \in \mathbb{Z}^*_p\}$$

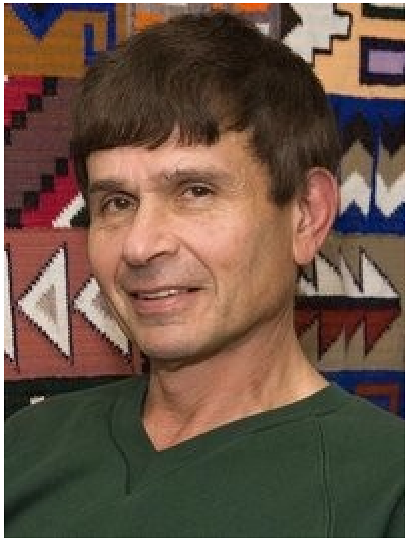
is a subgroup of $\mathbb{Z}^*_p$ of order q.

p is called safe prime if r=2

- Choosing uniform element in G?
  - Choose random h from $\mathbb{Z}^*_p$ and compute h^r mod p
- Determine if given h is in G (any h≠1 that is in G is a generator)
  - Check if $h^q = 1 \bmod p$

p and q need to be chosen such that the running time of the NFS (depends on the length of p), **and** the running time of generic algorithms (depends on the length of q) **will be approximately equal**.

Elliptic Curves



Neal Koblitz: **Elliptic Curve Cryptosystems**. Mathematics of Computation, AMS, 1987.

Victor S. Miller: **Use of Elliptic Curves in Cryptography**. Advances in Cryptology – CRYPTO ’85

- Groups discussed so far directly rely on modular arithmetic
- Why not use different groups? Elliptic curve groups?
  - Only generic algorithms for the DLP known!

Rationale: “it is extremely unlikely that an index calculus attack on the elliptic curve method will ever be able to work” [Miller, 85]

What are Elliptic Curves?

- An elliptic curve E over a field (we only condsider $\mathbb{F}_p$ with $p \geq 5$, and in particular large p) is a cubic equation

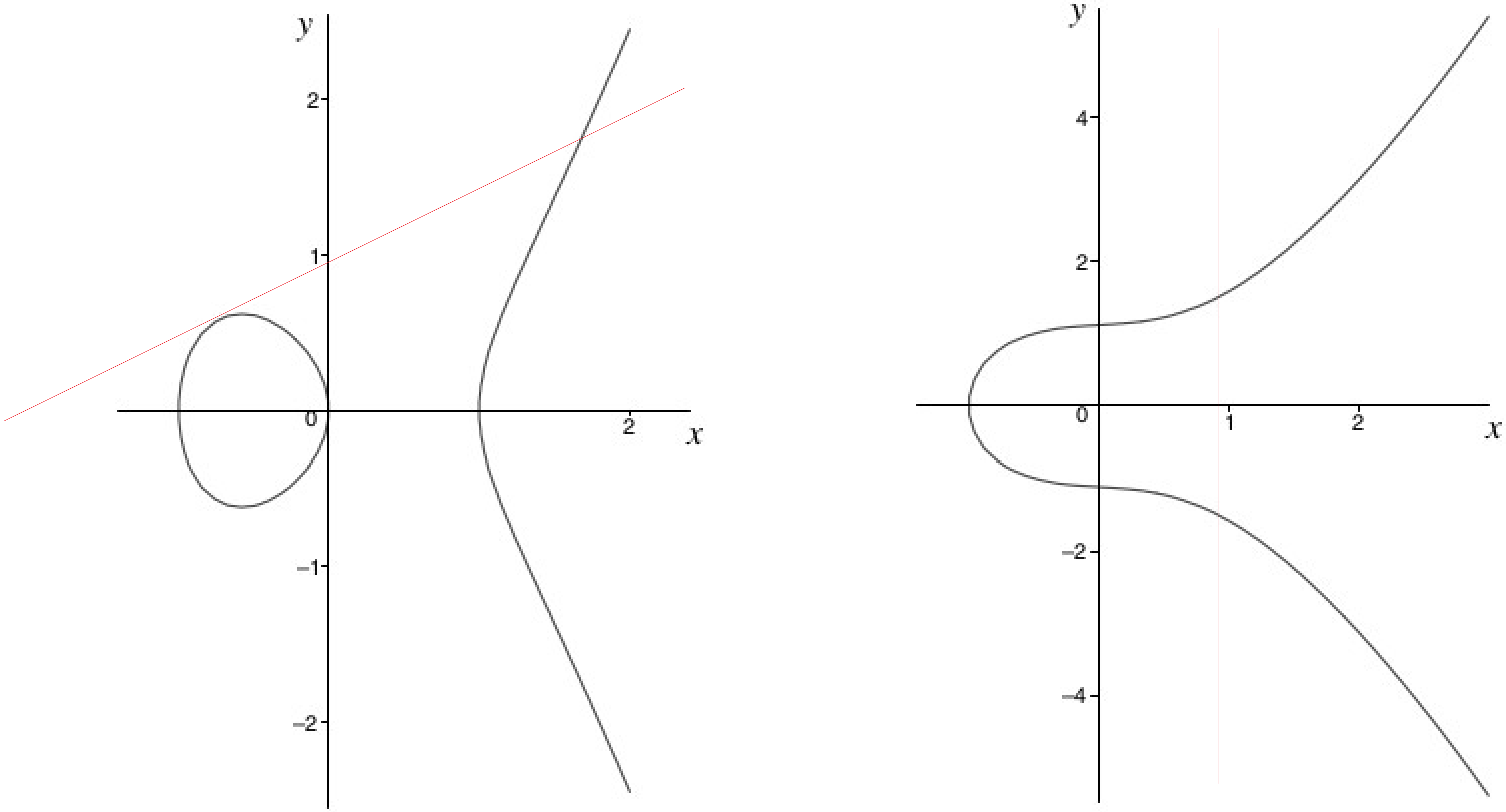
  $$y^2 = x^3 + ax + b \qquad \text{(short Weierstrass equation)}$$

  with $a, b \in \mathbb{Z}_p$ and $-16(4a^3 + 27b^2) \neq 0 \bmod p$ (the curve is “smooth”)

- Let $E(\mathbb{Z}_p) = \{(x, y) \mid x, y \in \mathbb{Z}_p \text{ and } y^2 = x^3 + ax + b \bmod p\} \cup \{\mathcal{O}\}$
  - The elements in $E(\mathbb{Z}_p)$ are called the points on the elliptic curve E
  - $\mathcal{O}$ is called the point at infinity (it will act as the identiy)

Elliptic Curves over the Reals

A useful way to think about E($\mathbb{Z}_p$) is to look at the graph over the reals



(a) $E_1 : y^2 = x^3 - x$

(b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

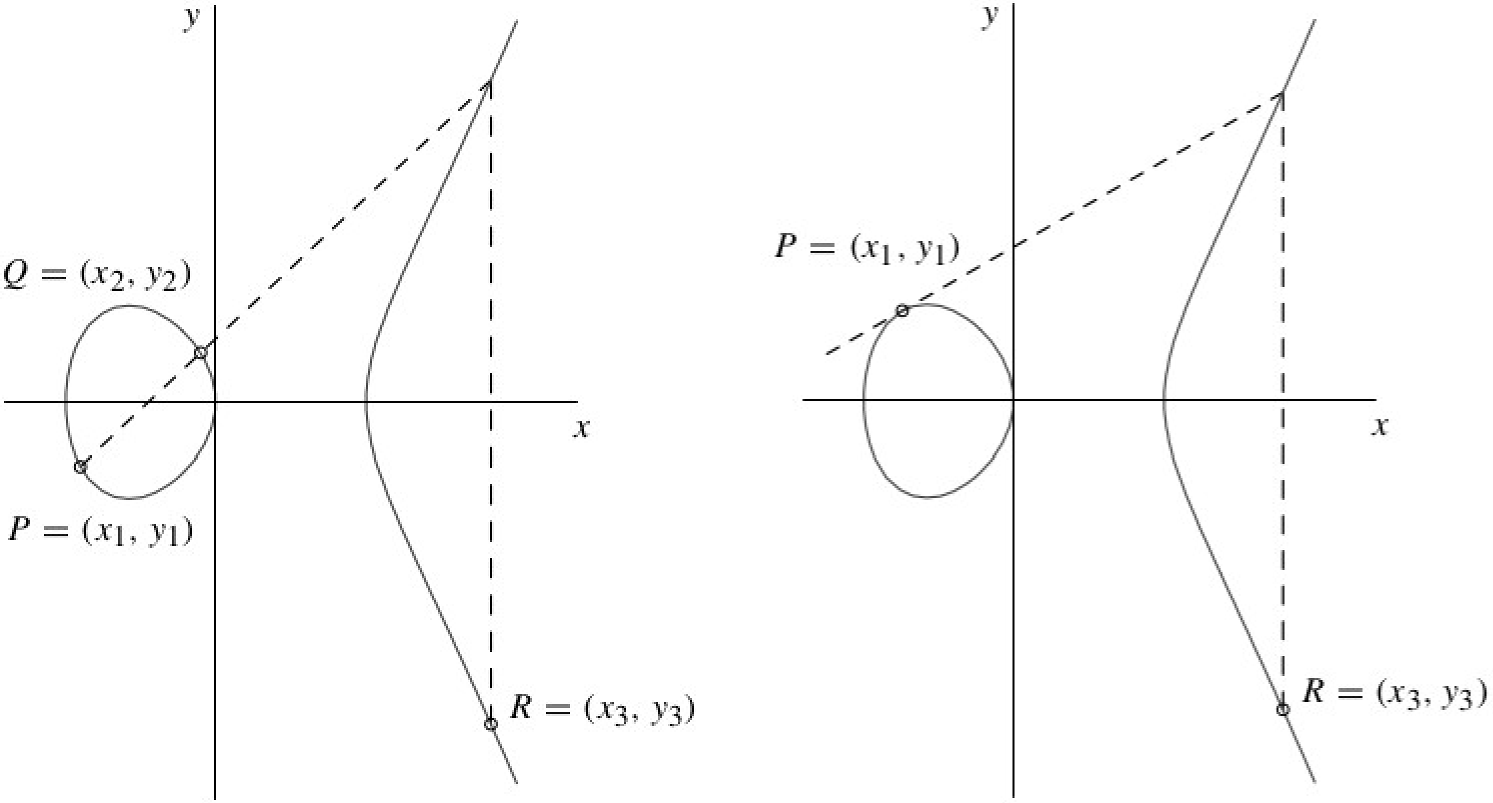
We can think of the point at infinity of sitting on top of the y-axis and lying on every vertical line

Every line intersecting the curve intersects in exactly three points

- Point P is counted twice if line is tangent to the curve
- Point at infinity is counted when the line is vertical

Elliptic Curves: Group Law (“chord-and-tangent rule”)

- $E(\mathbb{Z}_p)$ forms a group with additive identity $\mathcal{O}$
  - $\mathcal{O} + P = P + \mathcal{O} = P$ for all $P \in E(\mathbb{Z}_p)$
  - If $P = (x, y) \in E(\mathbb{Z}_p)$, then $(x, y) + (x, -y) = \mathcal{O}$ and $-\mathcal{O} = \mathcal{O}$



(a) Addition: $P + Q = R$.

(b) Doubling: $P + P = R$.

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 .$$

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{and} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 .$$

Elliptic Curves

- For cryptographic applications and in particular for the DLP to be hard we need (sub-) groups of large prime order.
- How large are these elliptic curve groups?
  - Let us define a quadratic residue (QR): An element y ∈ $\mathbb{Z}^*_p$ is a quadratic residue modulo p if there is an x ∈ $\mathbb{Z}^*_p$ such that $x^2 = y \bmod p$.
  - For p > 2 prime, half the elements in $\mathbb{Z}^*_p$ are QRs, and every QR has exactly two square roots.
  - If we look at the equation $y^2 = x^3 + ax + b$, each RHS value that is a QR yields two points on the curve and if RHS is 0 it yields one
  - So we heuristically expect to find expect to find $2 \cdot (p - 1)/2 + 1 = p$ points + the point of infinitey, i.e., p+1 points.

THEOREM 8.70 (Hasse bound): Let p be prime, and let E be an elliptic curve over $\mathbb{Z}_p$. Then $p + 1 - 2\sqrt{p} \leq |E(\mathbb{Z}_p)| \leq p + 1 + 2\sqrt{p}$.

Elliptic Curves

- How to find curves?
  - We could just randomly generate them: But for random curves the group order will be “close” to uniformly distributed in the Hasse interval
  - We also need to exclude weak curves, i.e., elliptic-curve groups over $\mathbb{Z}^*{}_p$ whose order is equal to p (anomalous curves) or p+1 (supersingular curves), etc.
  - There are efficient algorithms for counting points on curves, efficiently generating curves
- Typically we use pre-computed standardized curves
  - Standards for Efficient Cryptogrpahy (SEC)
  - National Institute of Standards and Technology (NIST)
  - ECC Brainpool (RFC 5639)
  - Curve25519, Curve448
  - Or BN or BLS if they need to be pairing-friendly

Elliptic Curves

- Now if we have a suitable elliptic curve group $E(\mathbb{Z}_p)$ (or a subgroup) of large prime order q generated by P, we can define the set {1P, ..., qP}
- We can define the elliptic curve DLP (ECDLP) as given Q=xP to compute $x \in \mathbb{Z}_q$
  - Analogously we can define CDH and DDH
- We can use our efficient square-and-multiply algorithm and apply it to this setting (double-and-add) to compute the scalar multiplication efficiently

Elliptic Curves

- Although curves standardized decades ago are still widely used, there happened a lot in the last decades
- Starting with Kocher'99, side-channel attacks and their counter-measures have become extremely sophisticated
- Decades of new research yielding faster, simpler and safer ways to do ECC
- Suspicion surrounding previous standards: Snowden leaks, dual EC-DRBG backdoor, etc., lead to conjectured weaknesses in the NIST curves
- Other specific classes of curves enable secure cryptographic pairings
  - and thus interesting applications such as practical identity- and attribute-based cryptography (see Guest Lecture)

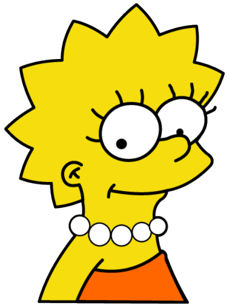
Back to Key Exchange Protocols

Example: KE in $\mathbb{Z}^*_p$ (128 bit security – p: 3072 bit)

p =

58096059953699580628595025333045743706869751763628952366614861522872037309971102257373360445331184072513261577549805174439905295945400471216628856 72187
03240103211163970644049884404985098905162720024476580704181239472968054002410482797658436938152229236120877904476989274322575173807697956881130957912 5
51133309324351955378481630638158016186020024749256844815024251530444957718760413642873858099017255157393414625583036640591500086964373205321856683254 5
29110790372283163413859958640669032595972518744716905954080501231020963901175074876001709536073423494575741627299485601330861695852995830467763701918 15
94088528345061285863898271763457294883546638879554311615446446330199254382340016292057090751175533888161918987295591531536698701292267685465517437915 79
08231548446347802601028917180324953960750418994855138111269773074789690748570437107161501213159220245567592412390131529197109564684063794429149416143571 0
7914462567329693649

g = 123456789



a =

7147687166405957187905360554739658269
2405186145916522354912615715297097100 6
7917003790492433011601949788108908769
6131592831386326210951294944584400497
4889298038584931918128447572321023987
1604390620061776483188754575562337708
5391250529236463183321912173214641346 5
5845254917228378772756695589845219962 2
0294508922696650742652691278024464164
0090259271040043389582611419862375878
9881936121879455918028640626798648395
7813927304368495559776413009721221824 9
1581096457937635455665546298837778595
6808915788215112735742204226463791705 9
9917677567304206984223924948169067778
9617492307207129760345580262107210922 0
5466273969774855354375899087960888262
7763290293452560094576029847391361388
7675543866224792652999780598864724145
3046219452761811989974647725290887806
0493179541951463829228890455778045929
4373052654104851802640020794151939838
5114342508427311982036827478946058710 0
3049774770692442789896899105721209635
77252034804024499138445834 48

g^a mod p =

1974966481832271932862620186142505559719097997625337606540081479948757754456670542185781051331382174972068905995549284294506678994 76
85466859559403409349363756245107893829696031348869617884814249135168725305460220296624704610577077157724832168211717424612832119567 8
53763152027864940346479735369199673699357709268717838560229887355895412105643052289961976145372708221782347574622380379001423505139 6
79904944650822466185016814995740147463845671662440190670139447244701505256941774637218509330253573938379198007057238142172902965163 9
30423436126876497170776348430066892397286870912166556866983097865780474015791661156350856988684748777267667120738609615294760711455 9
70634020905910370301818263552189873809454629455803556975259667634661469932774208847125574118475586611781220989551495243616019933653 2
60524221014748982566966601241957261004957255100220029328142187680601123107634554045672487613963996333449018578721192085185508037917 24

b =

6554562094646949336068268581603170 49
6942310472762446825117743874970612887 9
9577019369882685976279047911306230897
5863428283798589097017957365590672835
7138638957122466760949930089855480244
6403039544300748002507962036386619 31
5229886063541005322448463915897986412
1027377255837396548653931285483865070
9031919742048649235894391903529930326
7696100508840431979272991603892747747
4094858192679116146502863521484987 0
2328619342223917171215456861253006 72
0188085915004248494766867067840510
7153977068526645326383324039837473 3
7969702262426137716316320449382829 92
0398087034035751004673370850177483 8
8822224875309641791879395483731754 6
0348884930540399950519191679471224 05
3557093219350747155777569598163700 85
2039470528193639241108443600686183 5
4657249695621864372149726258332225 4
6599616046455854629937016589470425 2
6444562415789958697265293564785696709
2689604427965012098770368450012467927
6156391763995973638303866536272715 8

g^b mod p =

41160466206959330668322852565344187241077799922057207999357439723715636876203837833274247193966654496879381781932149526983361316993 7
98616481132079561694995740051820638531029247552928455062624713293012402770314013122096877114278839484659281611107827519695525804517 8
70525401646977350993692536199489589416306555110516192961313921978219875754298482646589345776888891556151450504809185615941297757604 9
07356322557280988097005839650171966585311010130843264742778656552512132877258716784203762419014390978793866584200569191199739672645 5
11075844855255374428884643379065403121253975718031032782719790076818413945341143157261205957499938963479817893107541948645774359056 73
17297003359658444520667122387439957656029195485616812623665738151941459294203701835123244046719122814558590904586127809180016633087 6
40732384471994880701268730488602792217616292819610462552195843277148172486262439624136130759567700180173857249994951177791494168821 88

g^{ab} mod p =

33016691952419214932376173359842624469122419995889465403633152639435009908862730297983333950118305919811398788006673 9
41999923137897071530703931787625845387670112454384952097943023330277750326501072451355120927957318323493435963669650 6
96832576948951102894369882151868949659775821854076751788583646416028947165136455249071396145660853601330164975397587 5
61065965575556747443818035795836022670874234817504556343707584096923082676703406111943765746699398938934828959960033 8
95037225133693267357174342882302601469923207111617139221959969109684671413364338274570937611250051430098365120196118 6
61346426768592656362458981725963724855810490365737198168441705399308267182734525284143333732542008838005923208917494 6
08653666498483604133403165043869263910628762715757575838312897105340103740703173150958280763950944870461798393013502 8
75965893832927519930791613188390431213291189300099481978999075869861089535914202794268747794235602210384 68

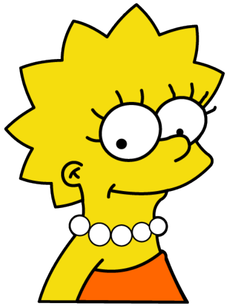
Example: KE using Elliptic Curves (128 bit security – p: 256 bit)

NIST Curve P-256

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

p = 115792089210356248762697446949407573530086143415290314195533631308867097853951

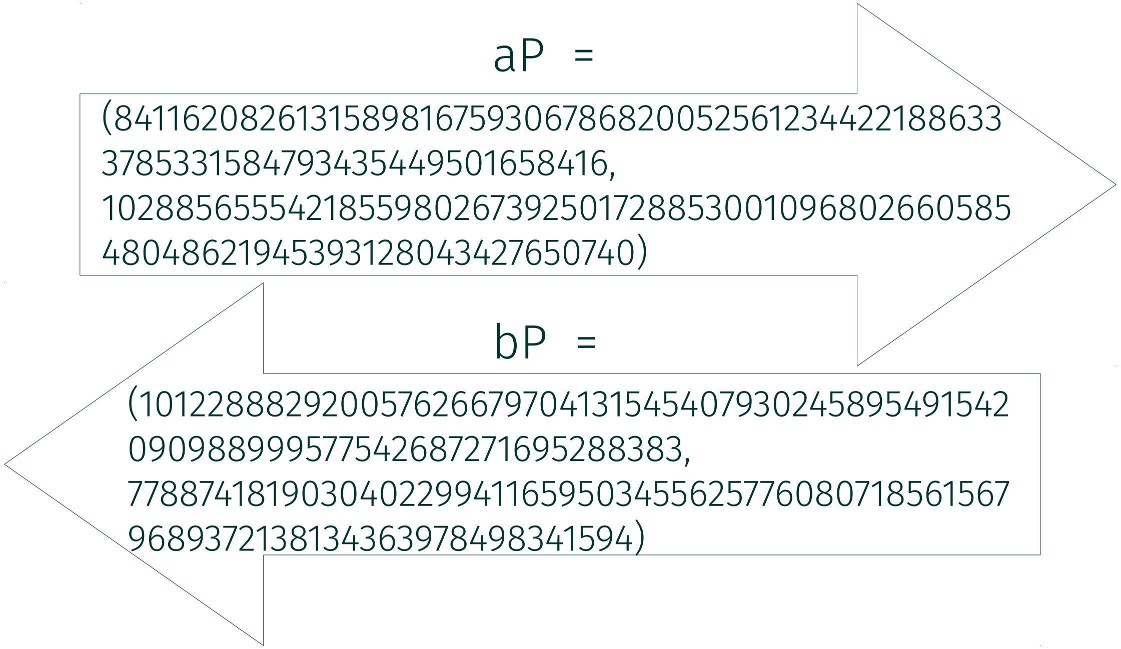
$$E(\mathbb{F}_p) : y^2 = x^3 - 3x + b$$



#E = 115792089210356248762697446949407573529996955224135760342422259061068512044369

P = (48439561293906451759052585252797914202762949526041747995844080717082404635286,
36134250956749795798585127919587881956611106672985015071877198253568414405109)

a=
89130644591246
03357763977064
14628550231450
28492835255603
183721922317324
614395



aP =

(84116208261315898167593067868200525612344221886333785331584793435449501658416,
102885655542185598026739250172885300109680266058548048621945393128043427650740)

bP =

(101228882920057626679704131545407930245895491542090988999577542687271695288383,
77887418190304022994116595034556257760807185615679689372138134363978498341594)

b=
10095557463932
78641880693831
61907080327719
10919058405391
67978108219340
5190826

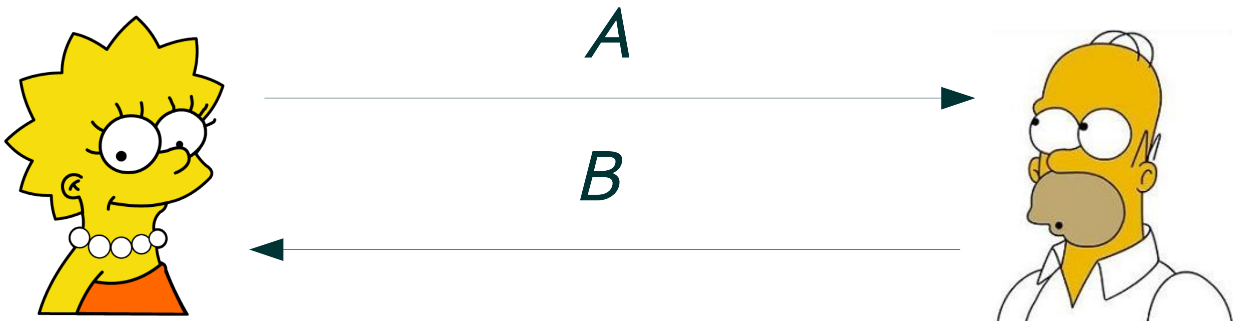
abP = (101228882920057626679704131545407930245895491542090988999577542687271695288383,
77887418190304022994116595034556257760807185615679689372138134363978498341594)

Diffie–Hellman(–Merkle) KE Protocol

- Now we are going to abstract away again the concrete setting and consider a group G of prime order q and generator g

$a \xleftarrow{\$} \mathbb{Z}_q; A \leftarrow g^a$

$b \xleftarrow{\$} \mathbb{Z}_p; B \leftarrow g^b$



$K_A \leftarrow B^a$

$K_B \leftarrow A^b$

Ok, how to prove security of this protocol?

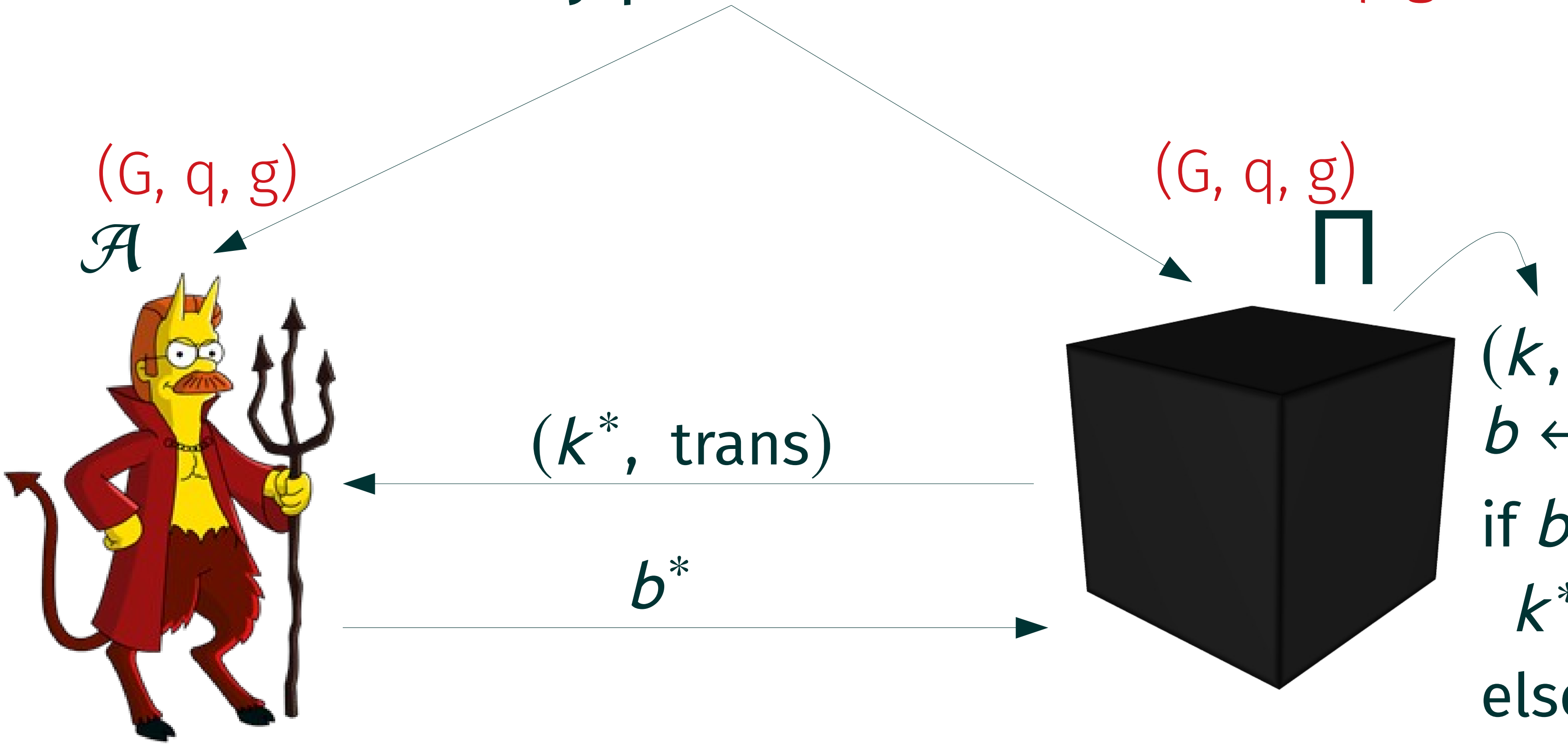
- Under DL? Other means of computing shared key?
- Under CHD? Only the complete shared key protected?
- Under DDH?

* definitional framework and idea of formulating assumptions not known back in the 70ies

Security Definition

$\widehat{\text{KE}}^{\text{eav}}_{\mathcal{A},\Pi}$ Security

security parameter $n \in \mathbb{N}$ $(G, q, g) \xleftarrow{\$} \mathcal{G}(1^n)$



(G, q, g)

$\mathcal{A}$

(G, q, g)

Π

$(k^*,\ \text{trans})$

b^*

```
(k, trans)
b ←$ {0, 1}
if b = 0
  k* := k
else  k* ←$ \overset{G}{\cancel{\{0, 1\}^n}}
```

A key-exchange protocol Π is secure in the presence of an eavesdropper if for every PPT adversary $\mathcal{A}$

$$Pr[b = b^*] \leq 1/2 + \text{negl}(n)$$

Analysis of the DH(M) KE Protocol

THEOREM 10.3: If the DDH problem is hard relative to G, then the Diffie–Hellman key-exchange protocol Π is secure in the presence of an eavesdropper (with respect to experiment $\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}$).

Proof: Let A be a PPT adversary.

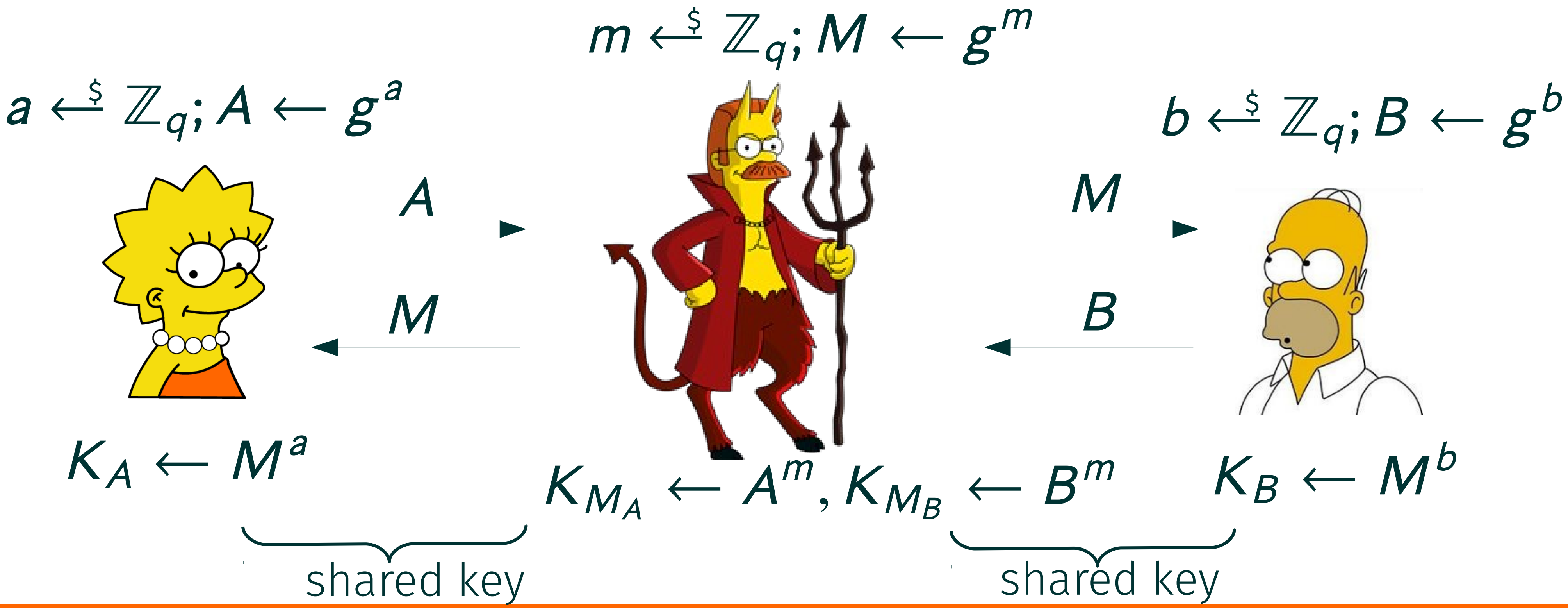
- Since Pr[b = 0] = Pr[b = 1] = ½, we have

$$
\begin{aligned}
&\Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \\
&= 1/2 \cdot \Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 | b = 0] + 1/2 \cdot \Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1 | b = 1] \\
&= 1/2 \cdot \Pr[\mathcal{A}(G, q, p, g^x, g^y, g^{xy}) = 0] + 1/2 \cdot \Pr[\mathcal{A}(G, q, p, g^x, g^y, g^z) = 1] \\
&= 1/2 \cdot (1 - \Pr[\mathcal{A}(G, q, p, g^x, g^y, g^{xy}) = 1]) + 1/2 \cdot \Pr[\mathcal{A}(G, q, p, g^x, g^y, g^z) = 1] \\
&= 1/2 + 1/2 \cdot (\Pr[\mathcal{A}(G, q, p, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(G, q, p, g^x, g^y, g^{xy}) = 1]) \\
&= 1/2 + 1/2 \cdot \underbrace{|\Pr[\mathcal{A}(G, q, p, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(G, q, p, g^x, g^y, g^{xy}) = 1]|}_{\leq \mathrm{negl}(n)} ,
\end{aligned}
$$

$$\Pr[\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + 1/2 \cdot \mathsf{negl}(n).$$

Analysis of the DH(M) KE Protocol

- Summary
  - Can prove eavesdropping security under DDH (not surprising; the assumption was basically modeled to abstract the analysis of these protocols)
- What did we miss so far?
  - Active adversaries: Man-in-the-middle



$m \xleftarrow{\$} \mathbb{Z}_q; M \leftarrow g^m$

$a \xleftarrow{\$} \mathbb{Z}_q; A \leftarrow g^a$

$b \xleftarrow{\$} \mathbb{Z}_q; B \leftarrow g^b$

$\xrightarrow{A}$ $\xleftarrow{M}$ $\xrightarrow{M}$ $\xleftarrow{B}$

$K_A \leftarrow M^a$

$K_{M_A} \leftarrow A^m, K_{M_B} \leftarrow B^m$

$K_B \leftarrow M^b$

shared key

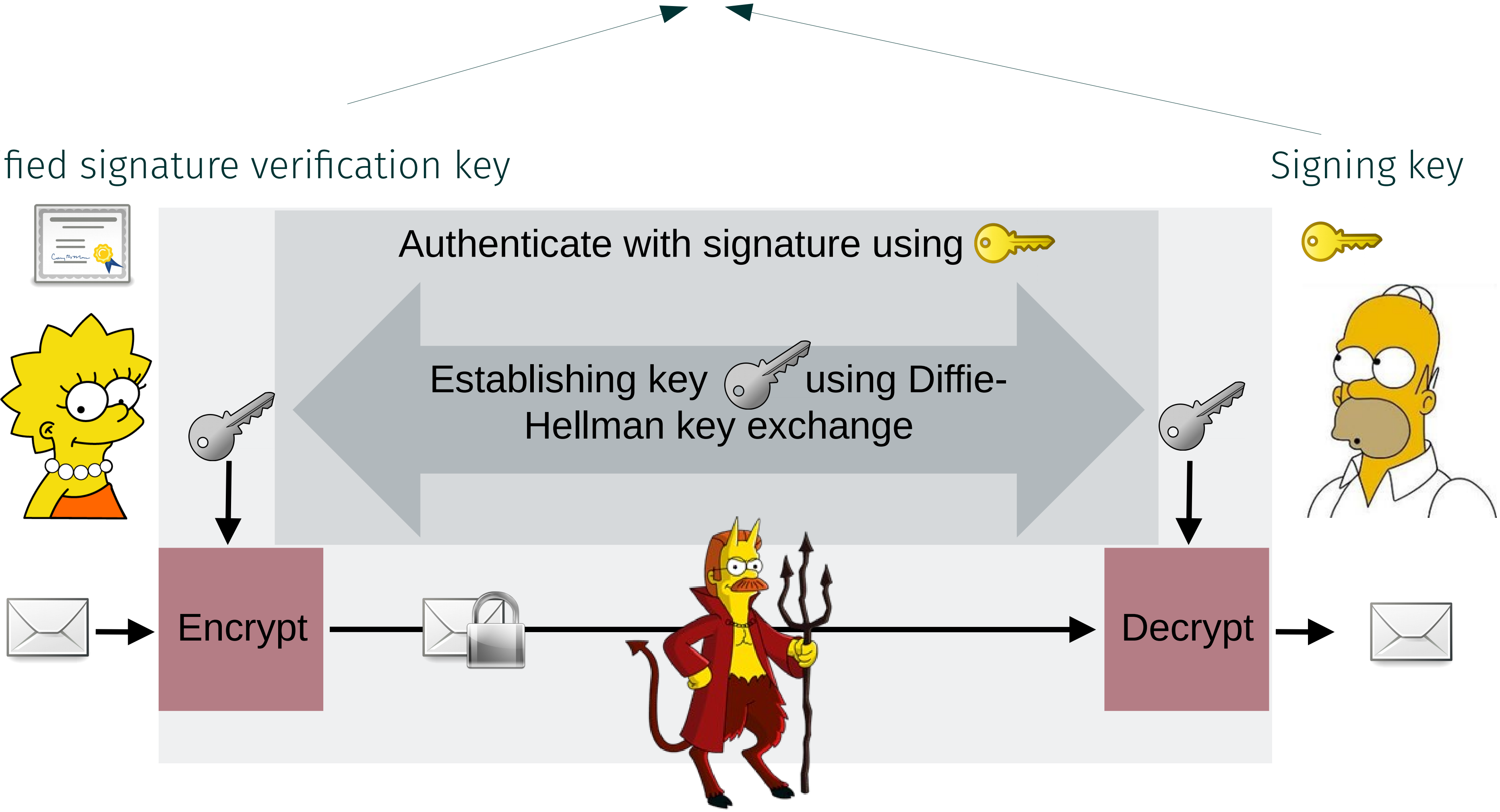
shared key

Countering man-in-the-middle attacks (Authenticated KE - AKE)

Will talk about signatures soon!

Certified signature verification key

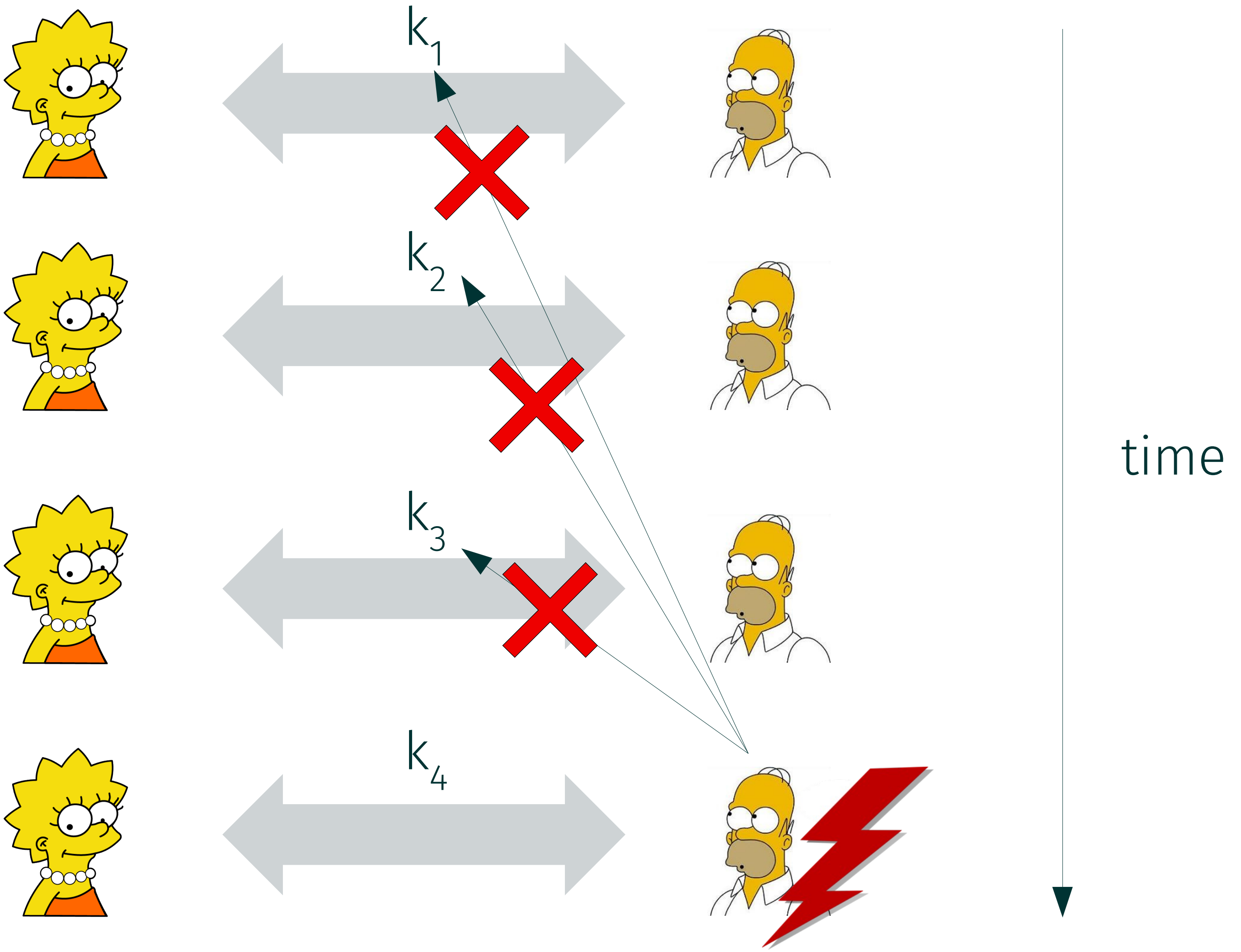
Signing key



Authenticate with signature using
Establishing key using Diffie-Hellman key exchange
Encrypt
Decrypt

Perfect Forward Secrecy

Another important property: Perfect forward secrecy



Alternatives to DL based KE Protocols: Outlook

Peter Shor

- Shor: computing discrete logarithms (and factoring) in polynomial time on a **quantum computer**
  - If we have a sufficiently powerful quantum computer, then DL and ECDL (as well as factoring) based systems will be dead
- What to do if this should happen?
  - Post-quantum cryptography: (asymmetric) cryptography that is conjectured to resists attacks using classical and quantum computers
- Very active field of research
  - Lattices
  - Codes
  - Isogenies (e.g., on supersingular elliptic curves – weak for EC crypto but good for PQ)
  - Etc.

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

https://csrc.nist.gov/projects/post-quantum-cryptography