Rainbow tables

Part 1: Reduction functions

- Hi, I'm Chris. I'm going to try to tell you about rainbow table reduction functions in 5 minutes.

Quick background

Rainbow tables

- Some quick background.

Quick background

Rainbow tables

Keys == Passwords

- First, whether I say “keys” or “passwords,” I'm referring to the same thing.

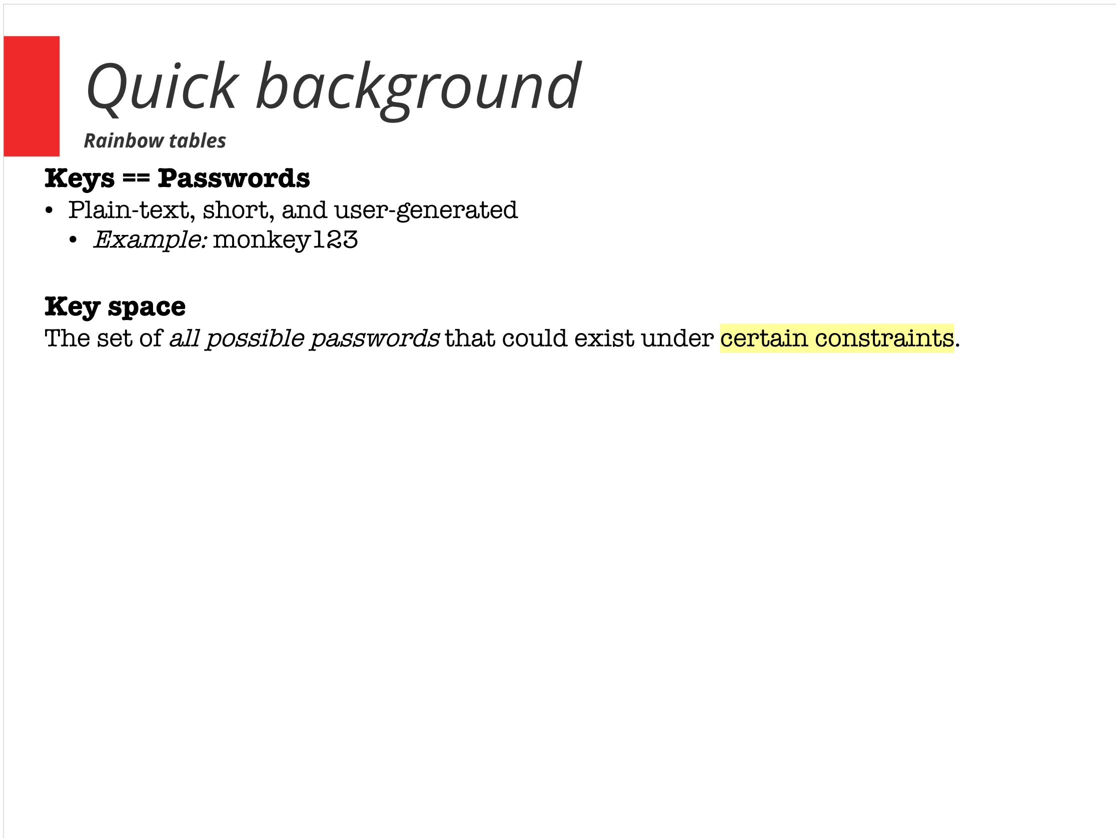
Quick background

Rainbow tables

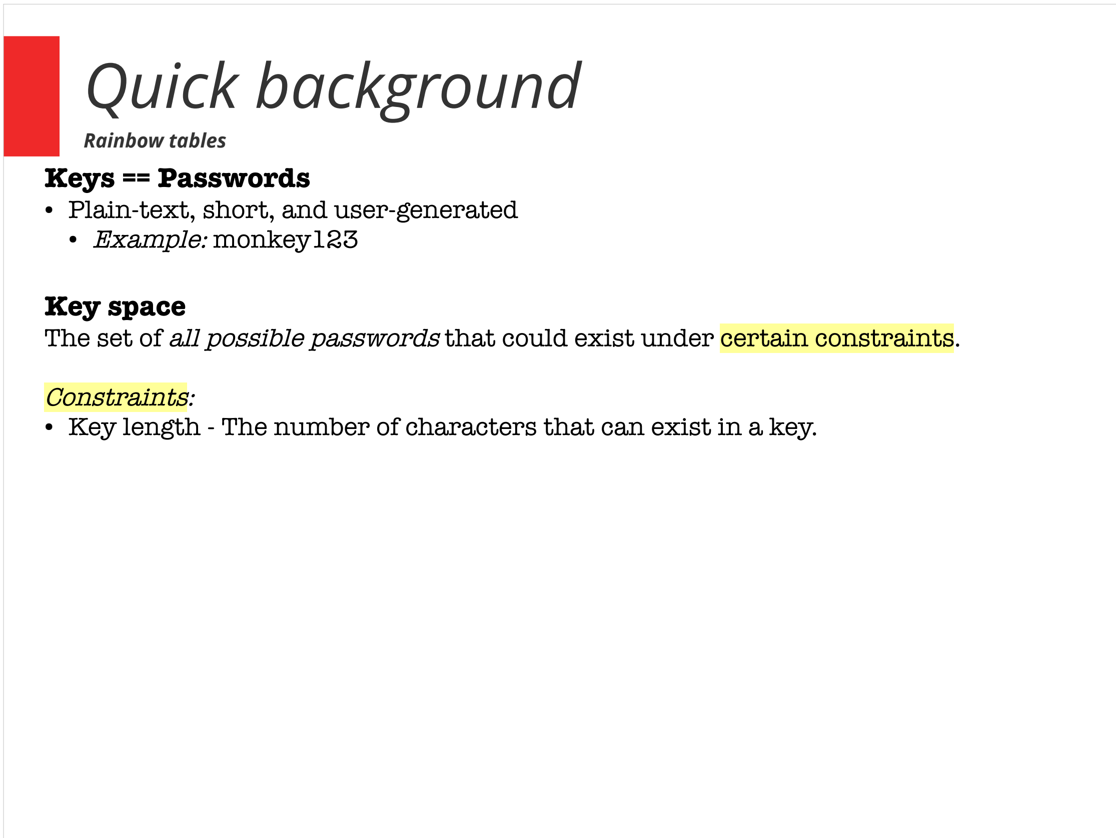
Keys == Passwords

- Plain-text, short, and user-generated
  - *Example:* monkey123

- They're plain-text strings.
- Short and user generated.



- Second, a “key space” is the set of all possible passwords that exist under certain constraints.



- Those constraints are:
  - “Key length” - how many characters a key, can be...

Quick background

Rainbow tables

Keys == Passwords

- Plain-text, short, and user-generated
  - *Example:* monkey123

Key space

The set of *all possible passwords* that could exist under certain constraints.

Constraints:

- Key length - The number of characters that can exist in a key.
- Allowable characters - Permissible characters in a key.
  - Often a-z, A-Z, 0-9. Sometimes symbols.

- … and, the “allowable characters” for a key.

Quick background

Rainbow tables

Keys == Passwords
- Plain-text, short, and user-generated
  - *Example:* monkey123

Key space
The set of *all possible passwords* that could exist under certain constraints.

Constraints:
- Key length - The number of characters that can exist in a key.
- Allowable characters - Permissible characters in a key.
  - Often a-z, A-Z, 0-9. Sometimes symbols.

`monkey123`: *Key length* = 9 / *allowable characters* = 36 (a-z0-9)

- For example, all numbers and lower-case letters represent 36 characters.
- With a key length of 9 and 36 allowable characters, we could create the password 'monkey123'

Quick background

Rainbow tables

Keys == Passwords

- Plain-text, short, and user-generated
  - *Example:* monkey123

Key space

The set of *all possible passwords* that could exist under certain constraints.

Constraints:

- Key length - The number of characters that can exist in a key.
- Allowable characters - Permissible characters in a key.
  - Often a-z, A-Z, 0-9. Sometimes symbols.

`monkey123`: *Key length* = 9 / *allowable characters* = 36 (a-z0-9)

Key space = ~36^9 = 101 trillion

- There are about 101 trillion possible passwords under these constraints.

Quick background

Rainbow tables

Keys == Passwords

- Plain-text, short, and user-generated
  - *Example:* monkey123

Key space

The set of *all possible passwords* that could exist under certain constraints.

Constraints:

- Key length - The number of characters that can exist in a key.
- Allowable characters - Permissible characters in a key.
  - Often a-z, A-Z, 0-9. Sometimes symbols.

`monkey123`: *Key length* = 9 / *allowable characters* = 36 (a-z0-9)

Key space = ~36^9 = 101 trillion

Cryptographic hash function - SHA-1

- Finally. A cryptographic hash function – SHA-1 in particular.

Quick background

Rainbow tables

Keys == Passwords

- Plain-text, short, and user-generated
  - *Example:* monkey123

Key space

The set of *all possible passwords* that could exist under certain constraints.

Constraints:

- Key length - The number of characters that can exist in a key.
- Allowable characters - Permissible characters in a key.
  - Often a-z, A-Z, 0-9. Sometimes symbols.

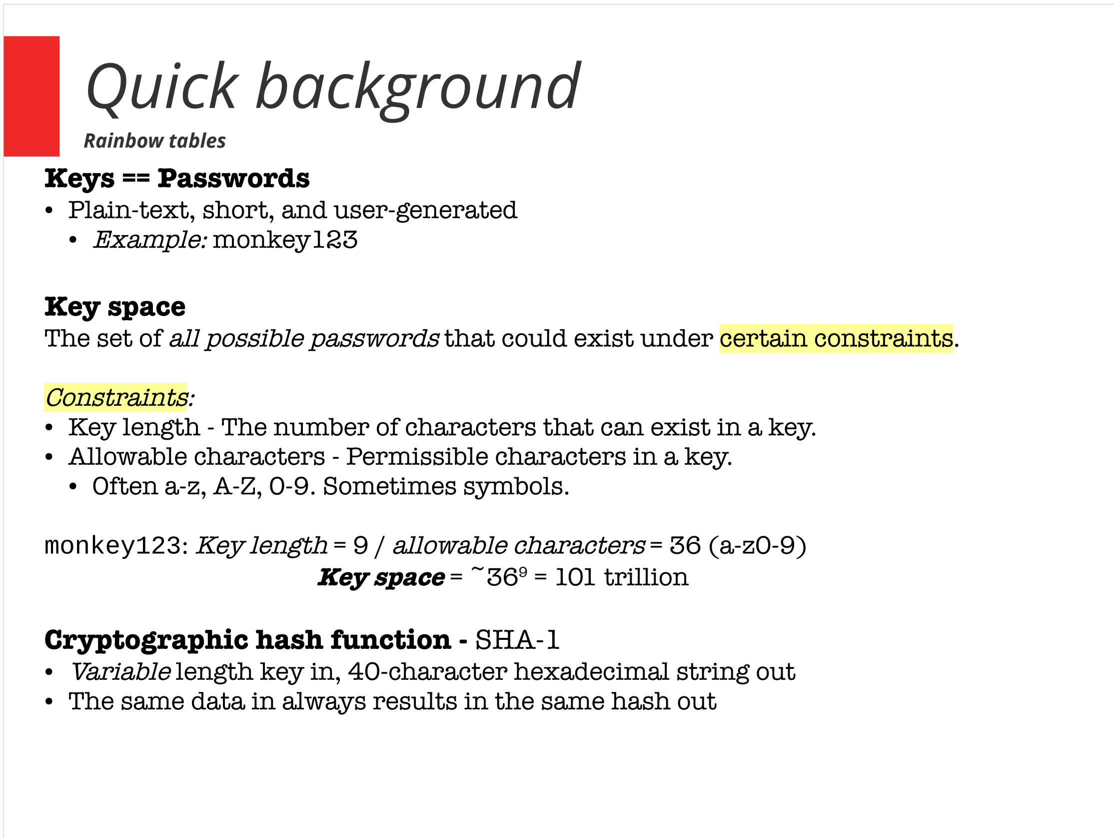
`monkey123`: *Key length* = 9 / *allowable characters* = 36 (a-z0-9)

Key space = ~36^9 = 101 trillion

Cryptographic hash function - SHA-1

- *Variable* length key in, 40-character hexadecimal string out

- With a hash function, any length of key in, and a 40-character hexadecimal string comes out.



- A second hash property is that the same key in always result in the same hash out.

X

Motivation

Rainbow tables

- Now, the motivation.

Motivation

Rainbow tables

You have a hash

`721d65122734734800a1edd6e68c03210e7b2aca`

- Say you have a hash.

Motivation

Rainbow tables

You have a hash
`721d65122734734800a1edd6e68c03210e7b2aca`

What password generated this hash?

- You'd like to know what plain-text password produced your hash.

Motivation

Rainbow tables

You have a hash

`721d65122734734800a1edd6e68c03210e7b2aca`

What password generated this hash?

- Most likely option: Brute force

- One way to answer this question is to use brute force.
- You would generate and hash every key in the key space, trying to find a hash that matches yours.

Motivation

Rainbow tables

You have a hash

`721d65122734734800a1edd6e68c03210e7b2aca`

What password generated this hash?

- Most likely option: Brute force
- Surprisingly quick

- This can be done surprisingly quickly.
- In a key space of 101 trillion, 'monkey123' might have taken about 2 hours to find.

Motivation

Rainbow tables

You have a hash

`721d65122734734800a1edd6e68c03210e7b2aca`

What password generated this hash?

- Most likely option: Brute force
- Surprisingly quick
- Not so quick when searching for multiple hashes

- But how long you were willing to wait for one hash, you may not be willing to wait for 100.

Motivation

Rainbow tables

You have a hash

`721d65122734734800a1edd6e68c03210e7b2aca`

What password generated this hash?

- Most likely option: Brute force
- Surprisingly quick
- Not so quick when searching for multiple hashes
  - Regenerating keys, re-hashing them

- Because for each hash you search for, you're regenerating and rehashing the same keys each time.

Motivation

Rainbow tables

You have a hash

`721d65122734734800a1edd6e68c03210e7b2aca`

What password generated this hash?

- Most likely option: Brute force
- Surprisingly quick
- Not so quick when searching for multiple hashes
  - Regenerating keys, re-hashing them
  - Saving this work isn't feasible

- Due to storage constraints, saving all key:hash pair results isn't an option.

X

Rainbow Tables

- This leads us to rainbow tables

Rainbow Tables

A rainbow table offers a method by which one can search a very large key space multiple times without actually saving or recomputing the entire key space for each search.

Rainbow tables are a tradeoff between computation time and size.

- A rainbow table offers a method by which one can search a very large key space multiple times, without the need to save, or recompute, the entire key space for each search.
- It is a computation time/size trade off.

X

Reduction function(s)

Rainbow tables

- Now, reduction functions!

Reduction function(s)

Rainbow tables

A reduction function takes a (*hash, salt*) pair and produces a plain-text key from the target key space.

- A reduction function takes as a pair, a hash, and a salt. It returns a plain-text key *that exists within the target key space*.

Reduction function(s)

Rainbow tables

A reduction function takes a (*hash, salt*) pair and produces a plain-text key from the target key space.

Consider it a “reverse hashing” function.

- It's *sort of* like a reverse hash function.

Reduction function(s)

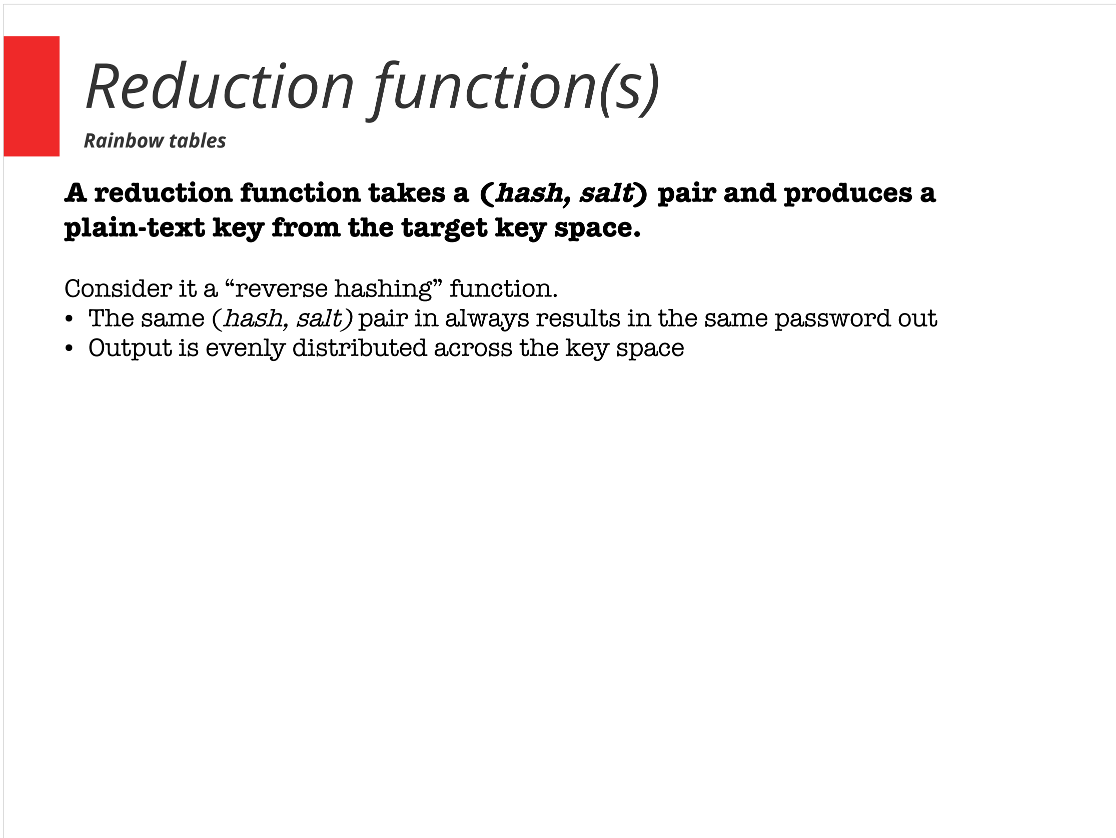
Rainbow tables

A reduction function takes a (*hash, salt*) pair and produces a plain-text key from the target key space.

Consider it a “reverse hashing” function.

- The same (*hash, salt)* pair in always results in the same password out

- It's deterministic, in that the same “hash:salt pair in”, always produces the same “plain-text key out”.



- And the entirety of the plain-text key space has equal chance of being produced by the reduction function.

Reduction function(s)

Rainbow tables

A reduction function takes a (*hash, salt*) pair and produces a plain-text key from the target key space.

Consider it a “reverse hashing” function.

- The same (*hash, salt)* pair in always results in the same password out
- Output is evenly distributed across the key space

But a reduction function does not produce the password that generated a given hash!

- But the plain-text key responsible for the input hash is *not generated* by the reduction function!

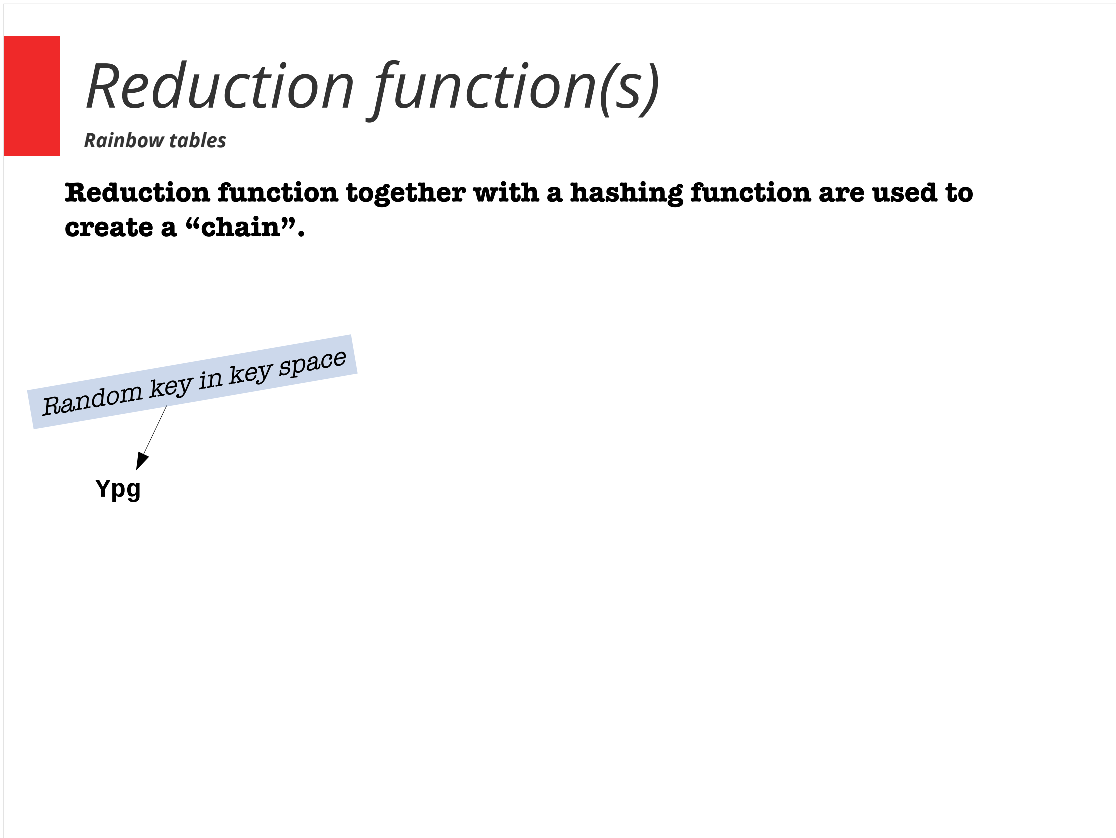
X

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

- Together, the reduction and hash function are used to create what's called a “chain.”



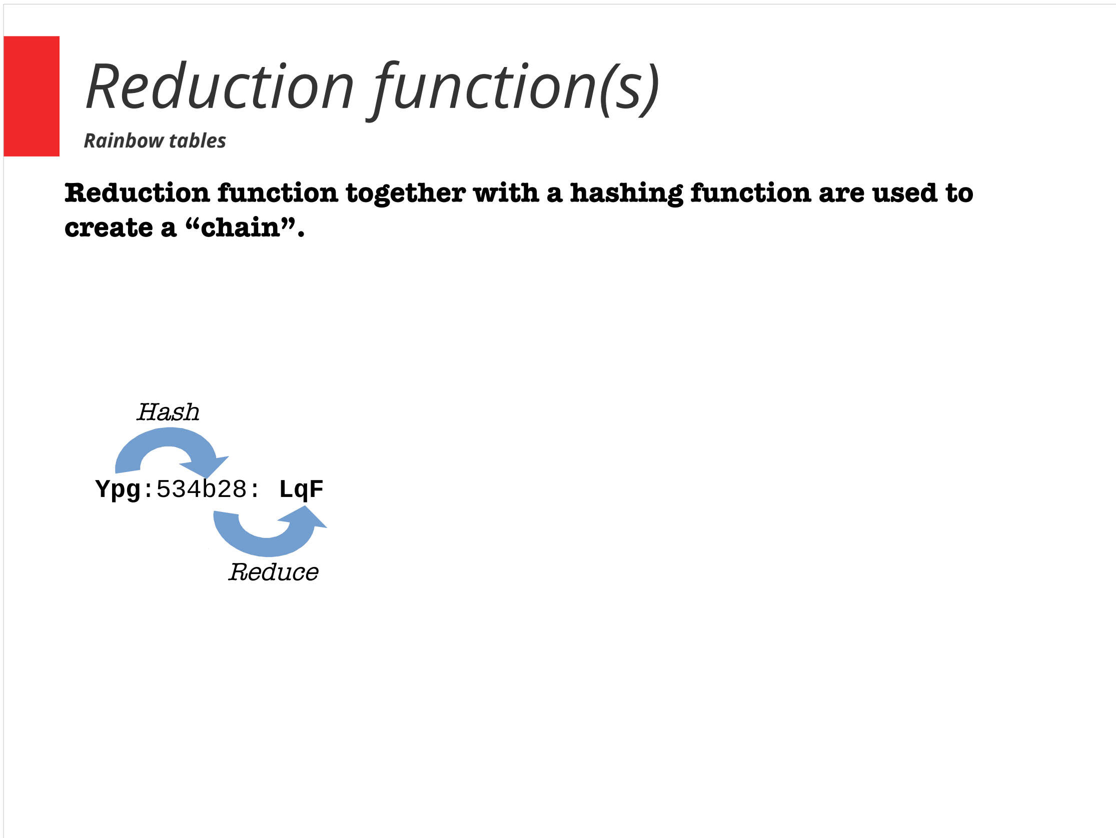
- To generate a chain, you start with a randomly generated plain-text key from the key space.

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a "chain".

- You hash that key.

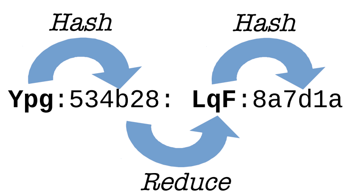


- Then use the reduction function on the resulting hash.

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

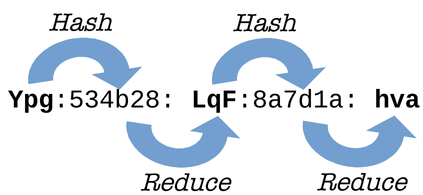


- This gives you a new key, which you hash...

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

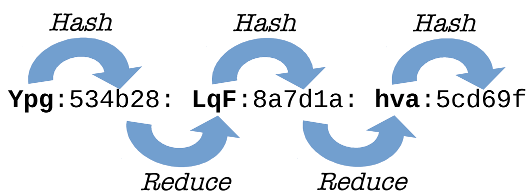


- And this carries on...

Reduction function(s)

Rainbow tables

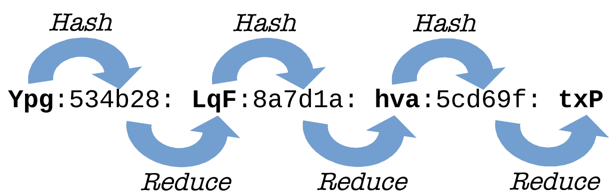
Reduction function together with a hashing function are used to create a "chain".



Reduction function(s)

Rainbow tables

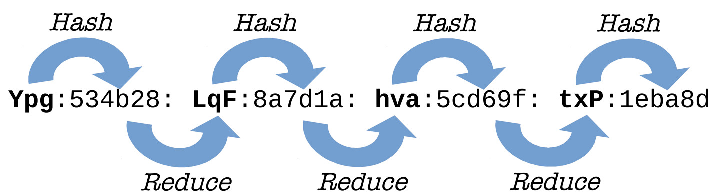
Reduction function together with a hashing function are used to create a “chain”.



Reduction function(s)

Rainbow tables

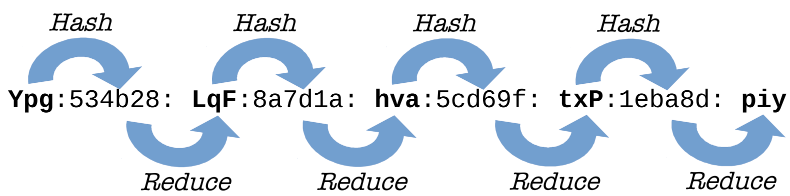
Reduction function together with a hashing function are used to create a “chain”.



Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.



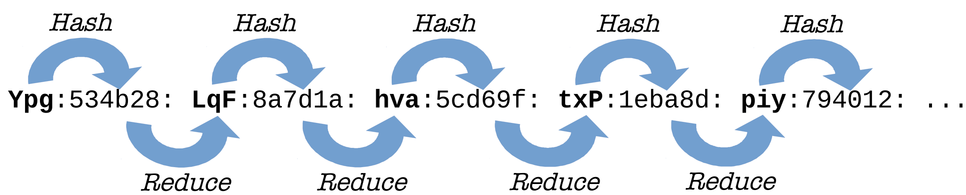
1

X

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a "chain".



- For an arbitrary number of iterations

Reduction function(s)

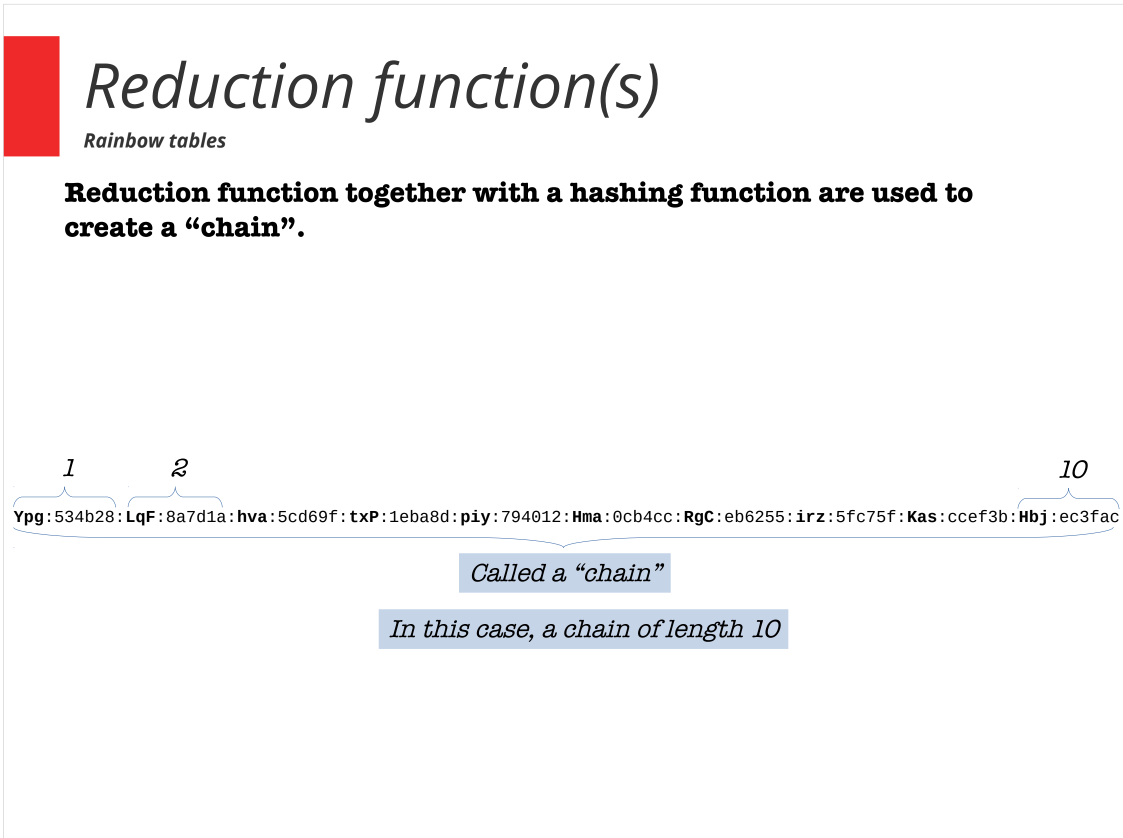
Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

Ypg:534b28:**LqF**:8a7d1a:**hva**:5cd69f:**txP**:1eba8d:**piy**:794012:**Hma**:0cb4cc:**RgC**:eb6255:**irz**:5fc75f:**Kas**:ccef3b:**Hbj**:ec3fac

Called a “chain”

- What you'll end up with is a “chain” of plain-text keys and hashes.



- This particular chain would be considered of length 10.

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a "chain".

In a Rainbow table, this chain constitutes one row of the table

Ypg:534b28:**LqF**:8a7d1a:**hva**:5cd69f:**txP**:1eba8d:**piy**:794012:**Hma**:0cb4cc:**RgC**:eb6255:**irz**:5fc75f:**Kas**:ccef3b:**Hbj**:ec3fac

- One chain is one row of a rainbow table.

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

In a Rainbow table, this chain constitutes one row of the table

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
```

- Now … a rainbow table will have millions of rows

Reduction function(s)

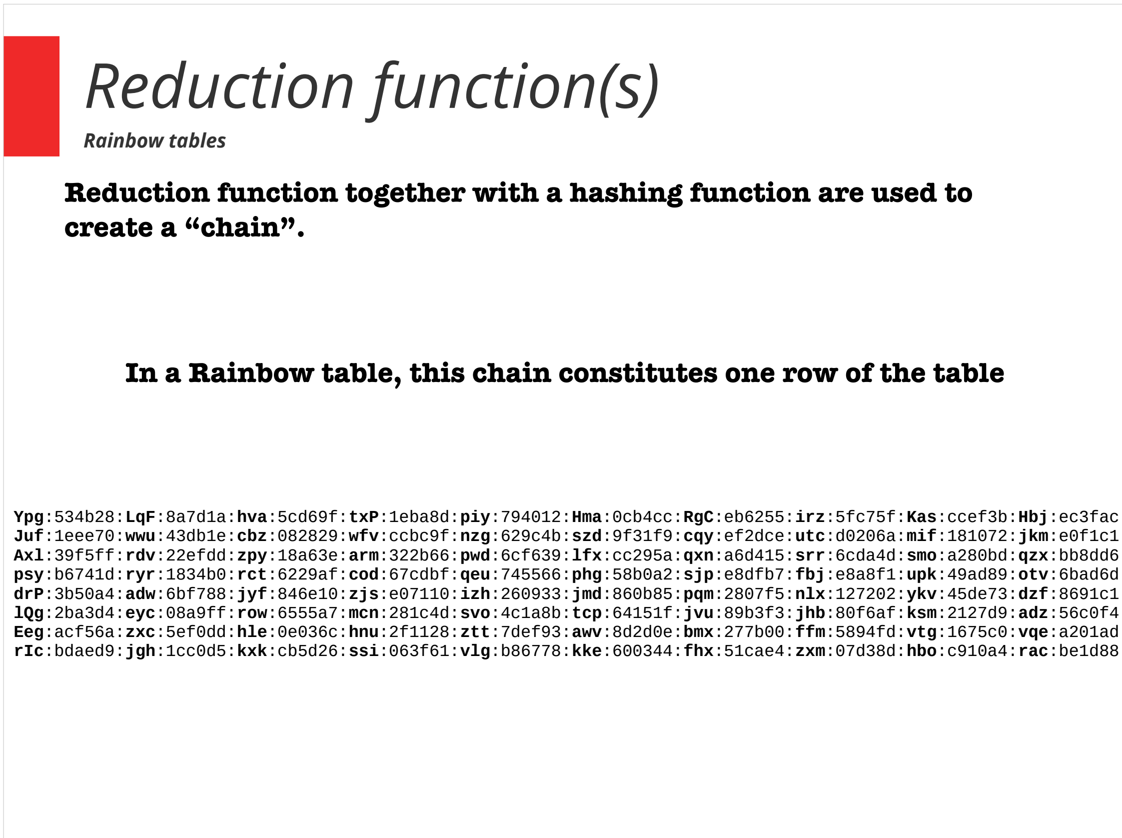
Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

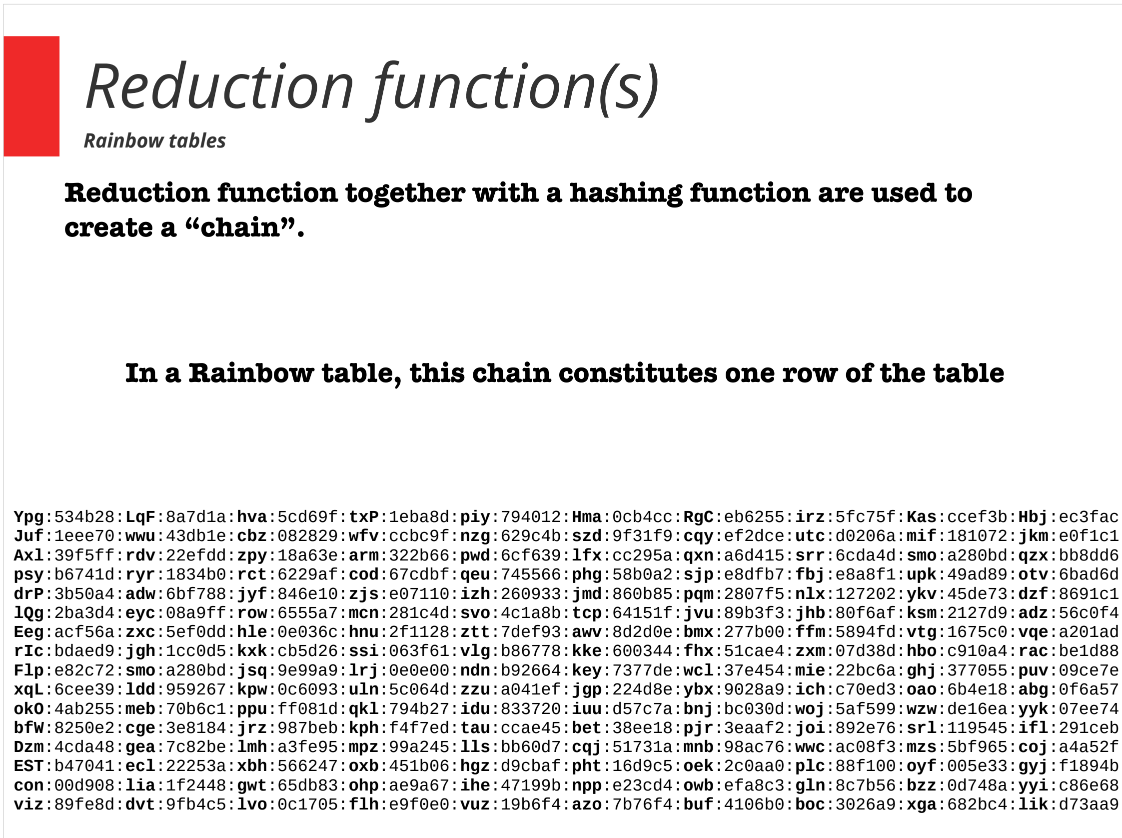
In a Rainbow table, this chain constitutes one row of the table

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sjp:e8dfb7:fbj:e8a8f1:upk:49ad89:otv:6bad6d
```

- Now a rainbow … table will have millions of rows



- Now a rainbow table will have millions … of rows



- Now a rainbow table will have millions of rows

Reduction function(s)

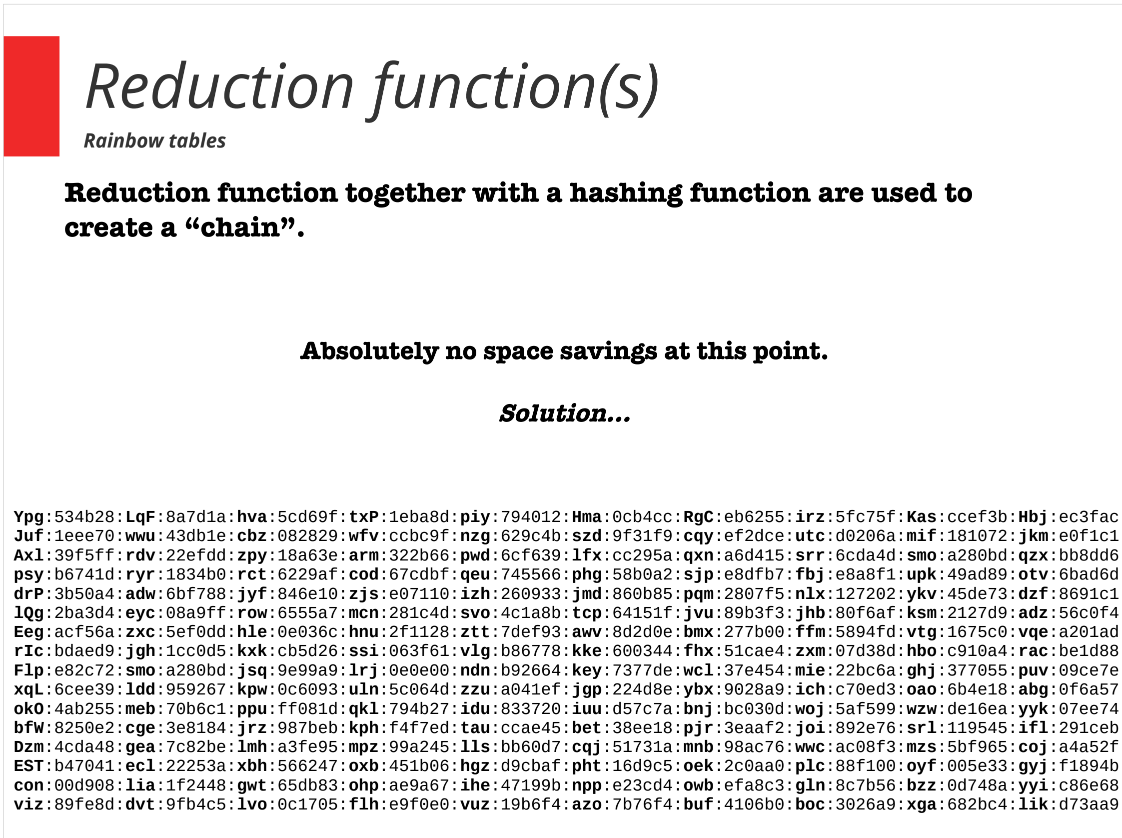
Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

Absolutely no space savings at this point.

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sjp:e8dfb7:fbj:e8a8f1:upk:49ad89:otv:6bad6d
drP:3b50a4:adw:6bf788:jyf:846e10:zjs:e07110:izh:260933:jmd:860b85:pqm:2807f5:nlx:127202:ykv:45de73:dzf:8691c1
lQg:2ba3d4:eyc:08a9ff:row:6555a7:mcn:281c4d:svo:4c1a8b:tcp:64151f:jvu:89b3f3:jhb:80f6af:ksm:2127d9:adz:56c0f4
Eeg:acf56a:zxc:5ef0dd:hle:0e036c:hnu:2f1128:ztt:7def93:awv:8d2d0e:bmx:277b00:ffm:5894fd:vtg:1675c0:vqe:a201ad
rIc:bdaed9:jgh:1cc0d5:kxk:cb5d26:ssi:063f61:vlg:b86778:kke:600344:fhx:51cae4:zxm:07d38d:hbo:c910a4:rac:be1d88
Flp:e82c72:smo:a280bd:jsq:9e99a9:lrj:0e0e00:ndn:b92664:key:7377de:wcl:37e454:mie:22bc6a:ghj:377055:puv:09ce7e
xqL:6cee39:ldd:959267:kpw:0c6093:uln:5c064d:zzu:a041ef:jgp:224d8e:ybx:9028a9:ich:c70ed3:oao:6b4e18:abg:0f6a57
okO:4ab255:meb:70b6c1:ppu:ff081d:qkl:794b27:idu:833720:iuu:d57c7a:bnj:bc030d:woj:5af599:wzw:de16ea:yyk:07ee74
bfW:8250e2:cge:3e8184:jrz:987beb:kph:f4f7ed:tau:ccae45:bet:38ee18:pjr:3eaaf2:joi:892e76:srl:119545:ifl:291ceb
Dzm:4cda48:gea:7c82be:lmh:a3fe95:mpz:99a245:lls:bb60d7:cqj:51731a:mnb:98ac76:wwc:ac08f3:mzs:5bf965:coj:a4a52f
EST:b47041:ecl:22253a:xbh:566247:oxb:451b06:hgz:d9cbaf:pht:16d9c5:oek:2c0aa0:plc:88f100:oyf:005e33:gyj:f1894b
con:00d908:lia:1f2448:gwt:65db83:ohp:ae9a67:ihe:47199b:npp:e23cd4:owb:efa8c3:gln:8c7b56:bzz:0d748a:yyi:c86e68
viz:89fe8d:dvt:9fb4c5:lvo:0c1705:flh:e9f0e0:vuz:19b6f4:azo:7b76f4:buf:4106b0:boc:3026a9:xga:682bc4:lik:d73aa9
```

- But, as shown here, there is absolutely no space savings over just saving every key:hash pair from a key space.



- The solution...

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a “chain”.

Absolutely no space savings at this point.

Solution…

Drop all entries from the middle of the table

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sjp:e8dfb7:fbj:e8a8f1:upk:49ad89:otv:6bad6d
drP:3b50a4:adw:6bf788:jyf:846e10:zjs:e07110:izh:260933:jmd:860b85:pqm:2807f5:nlx:127202:ykv:45de73:dzf:8691c1
lQg:2ba3d4:eyc:08a9ff:row:6555a7:mcn:281c4d:svo:4c1a8b:tcp:64151f:jvu:89b3f3:jhb:80f6af:ksm:2127d9:adz:56c0f4
Eeg:acf56a:zxc:5ef0dd:hle:0e036c:hnu:2f1128:ztt:7def93:awv:8d2d0e:bmx:277b00:ffm:5894fd:vtg:1675c0:vqe:a201ad
rIc:bdaed9:jgh:1cc0d5:kxk:cb5d26:ssi:063f61:vlg:b86778:kke:600344:fhx:51cae4:zxm:07d38d:hbo:c910a4:rac:be1d88
Flp:e82c72:smo:a280bd:jsq:9e99a9:lrj:0e0e00:ndn:b92664:key:7377de:wcl:37e454:mie:22bc6a:ghj:377055:puv:09ce7e
xqL:6cee39:ldd:959267:kpw:0c6093:uln:5c064d:zzu:a041ef:jgp:224d8e:ybx:9028a9:ich:c70ed3:oao:6b4e18:abg:0f6a57
okO:4ab255:meb:70b6c1:ppu:ff081d:qkl:794b27:idu:833720:iuu:d57c7a:bnj:bc030d:woj:5af599:wzw:de16ea:yyk:07ee74
bfW:8250e2:cge:3e8184:jrz:987beb:kph:f4f7ed:tau:ccae45:bet:38ee18:pjr:3eaaf2:joi:892e76:srl:119545:ifl:291ceb
Dzm:4cda48:gea:7c82be:lmh:a3fe95:mpz:99a245:lls:bb60d7:cqj:51731a:mnb:98ac76:wwc:ac08f3:mzs:5bf965:coj:a4a52f
EST:b47041:ecl:22253a:xbh:566247:oxb:451b06:hgz:d9cbaf:pht:16d9c5:oek:2c0aa0:plc:88f100:oyf:005e33:gyj:f1894b
con:00d908:lia:1f2448:gwt:65db83:ohp:ae9a67:ihe:47199b:npp:e23cd4:owb:efa8c3:gln:8c7b56:bzz:0d748a:yyi:c86e68
viz:89fe8d:dvt:9fb4c5:lvo:0c1705:flh:e9f0e0:vuz:19b6f4:azo:7b76f4:buf:4106b0:boc:3026a9:xga:682bc4:lik:d73aa9
```

- Is to drop the entire middle of the table

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a "chain".

Absolutely no space savings at this point.

Solution…

Drop all entries from the middle of the table

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sjp:e8dfb7:fbj:e8a8f1:upk:49ad89:otv:6bad6d
drP:3b50a4:adw:6bf788:jyf:846e10:zjs:e07110:izh:260933:jmd:860b85:pqm:2807f5:nlx:127202:ykv:45de73:dzf:8691c1
lQg:2ba3d4:eyc:08a9ff:row:6555a7:mcn:281c4d:svo:4c1a8b:tcp:64151f:jvu:89b3f3:jhb:80f6af:ksm:2127d9:adz:56c0f4
Eeg:acf56a:zxc:5ef0dd:hle:0e036c:hnu:2f1128:ztt:7def93:awv:8d2d0e:bmx:277b00:ffm:5894fd:vtg:1675c0:vqe:a201ad
rIc:bdaed9:jgh:1cc0d5:kxk:cb5d26:ssi:063f61:vlg:b86778:kke:600344:fhx:51cae4:zxm:07d38d:hbo:c910a4:rac:be1d88
Flp:e82c72:smo:a280bd:jsq:9e99a9:lrj:0e0e00:ndn:b92664:key:7377de:wcl:37e454:mie:22bc6a:ghj:377055:puv:09ce7e
xqL:6cee39:ldd:959267:kpw:0c6093:uln:5c064d:zzu:a041ef:jgp:224d8e:ybx:9028a9:ich:c70ed3:oao:6b4e18:abg:0f6a57
okO:4ab255:meb:70b6c1:ppu:ff081d:qkl:794b27:idu:833720:iuu:d57c7a:bnj:bc030d:woj:5af599:wzw:de16ea:yyk:07ee74
bfW:8250e2:cge:3e8184:jrz:987beb:kph:f4f7ed:tau:ccae45:bet:38ee18:pjr:3eaaf2:joi:892e76:srl:119545:ifl:291ceb
Dzm:4cda48:gea:7c82be:lmh:a3fe95:mpz:99a245:lls:bb60d7:cqj:51731a:mnb:98ac76:wwc:ac08f3:mzs:5bf965:coj:a4a52f
EST:b47041:ecl:22253a:xbh:566247:oxb:451b06:hgz:d9cbaf:pht:16d9c5:oek:2c0aa0:plc:88f100:oyf:005e33:gyj:f1894b
con:00d908:lia:1f2448:gwt:65db83:ohp:ae9a67:ihe:47199b:npp:e23cd4:owb:efa8c3:gln:8c7b56:bzz:0d748a:yyi:c86e68
viz:89fe8d:dvt:9fb4c5:lvo:0c1705:flh:e9f0e0:vuz:19b6f4:azo:7b76f4:buf:4106b0:boc:3026a9:xga:682bc4:lik:d73aa9
```

- From the first key on the left, to the final hash on the right.

X

Reduction function(s)

Rainbow tables

Reduction function together with a hashing function are used to create a "chain".

Absolutely no space savings at this point.

Solution…

Drop all entries from the middle of the table

```
Ypg:ec3fac
Juf:e0f1c1
Axl:bb8dd6
psy:6bad6d
drP:8691c1
lQg:56c0f4
Eeg:a201ad
rIc:be1d88
Flp:09ce7e
xqL:0f6a57
okO:07ee74
bfW:291ceb
Dzm:a4a52f
EST:f1894b
con:c86e68
viz:d73aa9
```

- Keeping only the first key, and final hash.

XX

Searching

Rainbow tables

- There's an obvious question at this point...

Searching

Rainbow tables

How do we search this table in such a way that we can find the any of the 144 key:hash pairs dropped from the full table?

```
Ypg:ec3fac
Juf:e0f1c1
Axl:bb8dd6
psy:6bad6d
drP:8691c1
lQg:56c0f4
Eeg:a201ad
rIc:be1d88
Flp:09ce7e
xqL:0f6a57
okO:07ee74
bfW:291ceb
Dzm:a4a52f
EST:f1894b
con:c86e68
viz:d73aa9
```

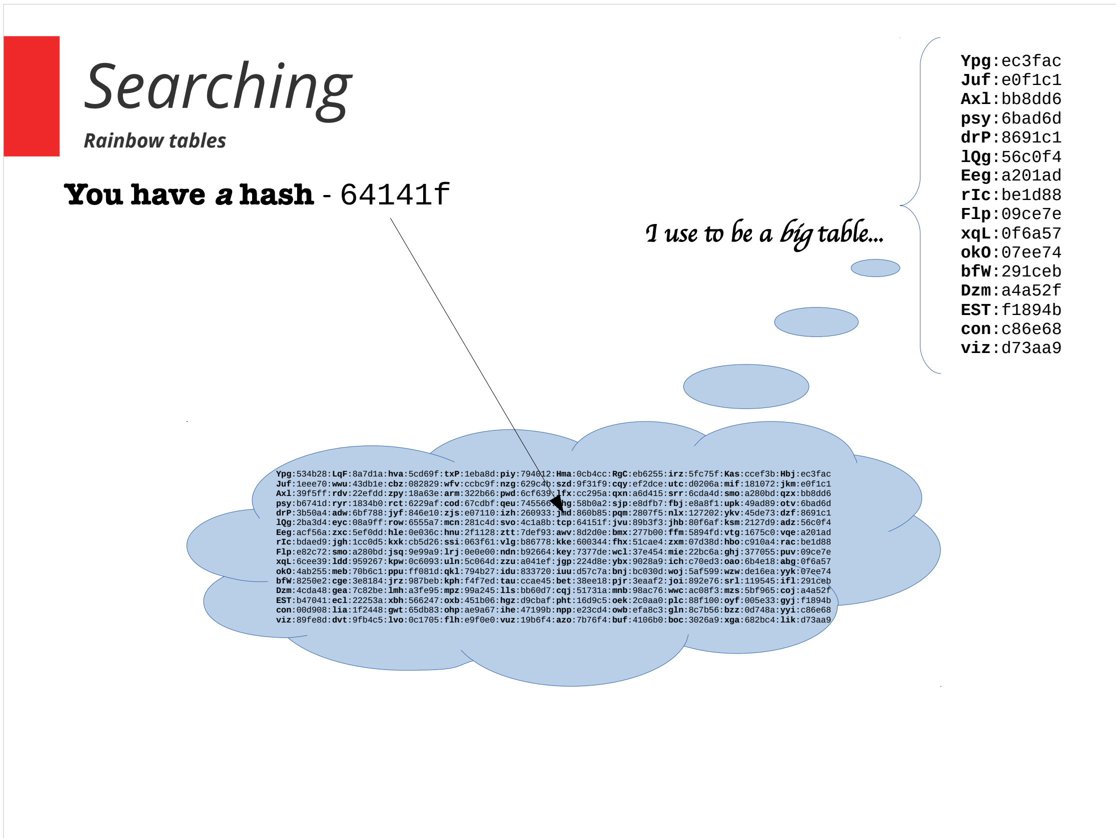
- How can we search this reduced table and still find any of the key:hash pairs that were dropped from the full table?

Searching

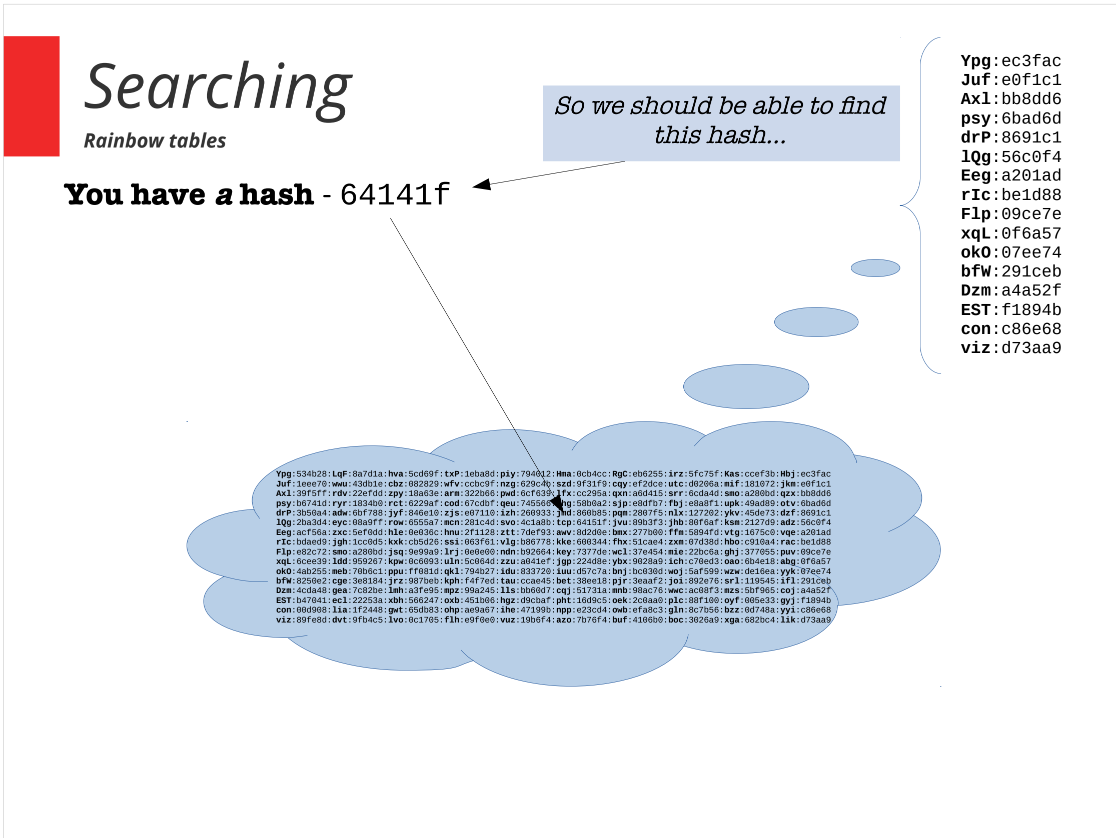
Rainbow tables

You have *a* hash - `64141f`

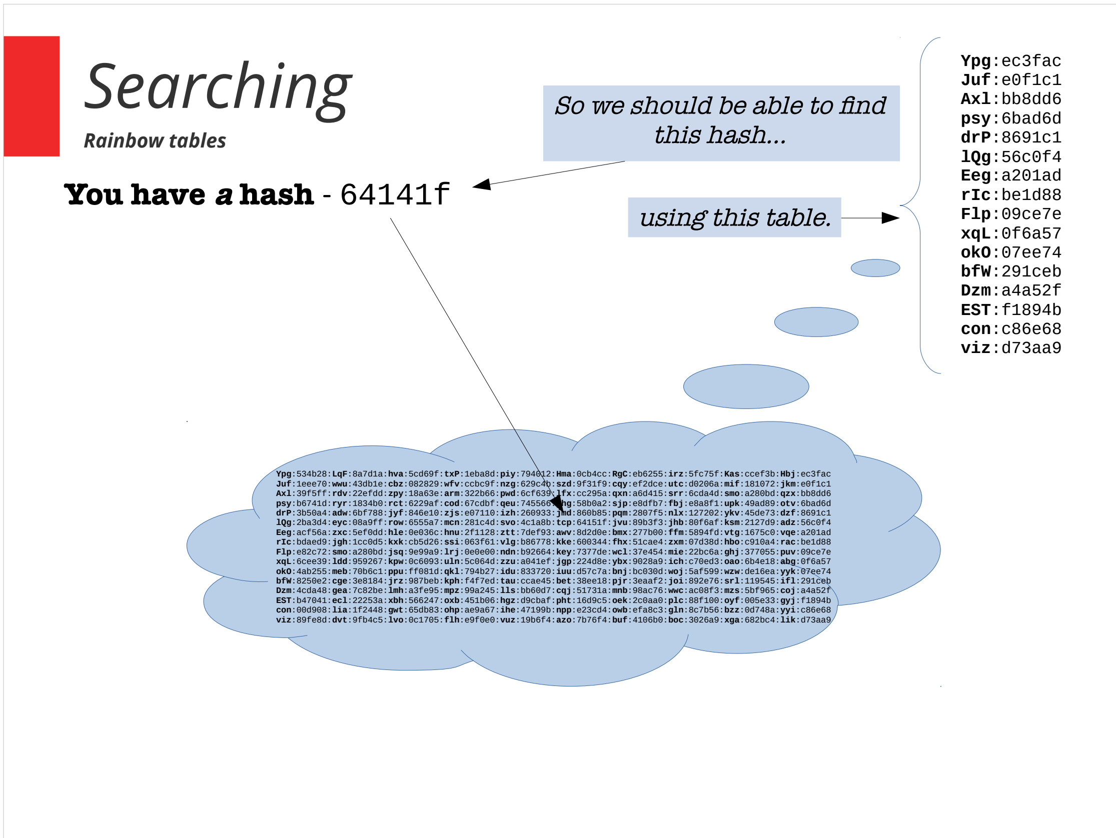
- Well lets see. Suppose you have a hash...



- A hash from a key:hash pair that was computed in the full-sized table, but was dropped.



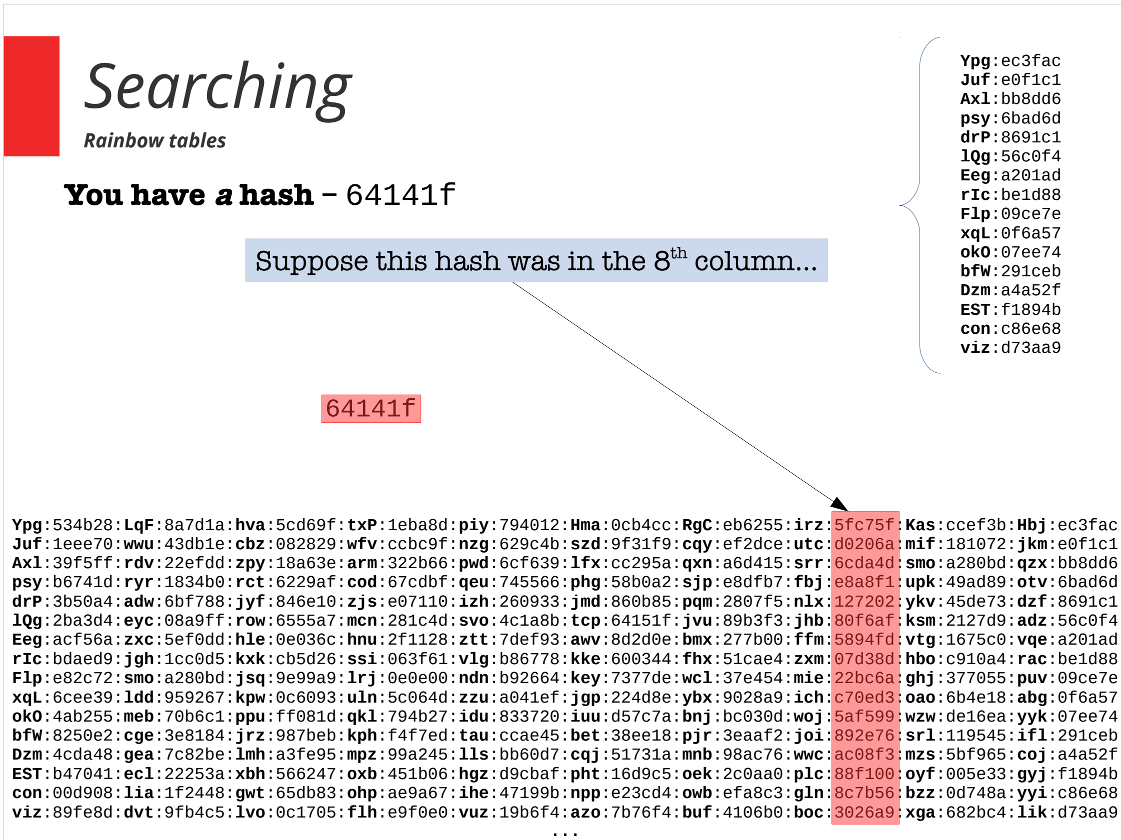
- We should be able to find this hash...



- … in our reduced-size table.

X

- But in this case, it's not in there.



- But suppose when the larger table was generated initially, that *our* hash was in, as an example, the 8^{th} column

Searching

Rainbow tables

You have *a* hash – 64141f

Suppose this hash was in the 8th column...

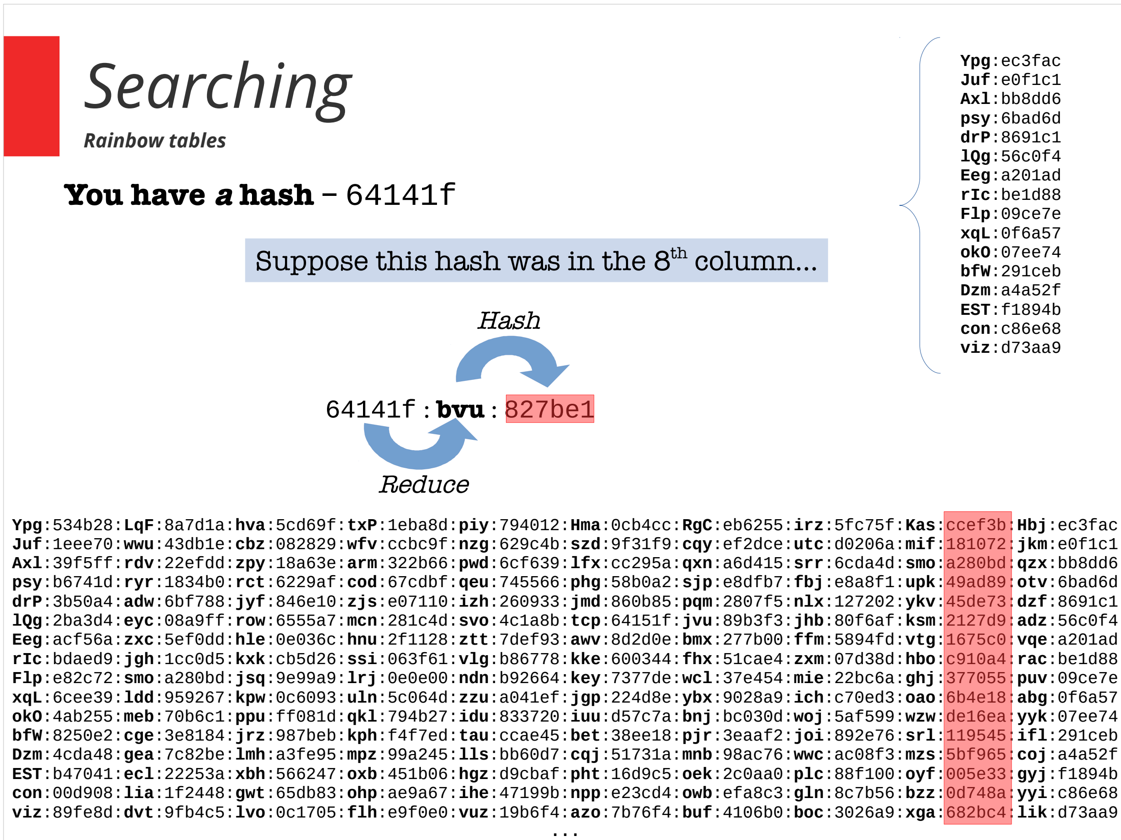
```
Ypg:ec3fac
Juf:e0f1c1
Axl:bb8dd6
psy:6bad6d
drP:8691c1
lQg:56c0f4
Eeg:a201ad
rIc:be1d88
Flp:09ce7e
xqL:0f6a57
okO:07ee74
bfW:291ceb
Dzm:a4a52f
EST:f1894b
con:c86e68
viz:d73aa9
```

64141f : **bvu**

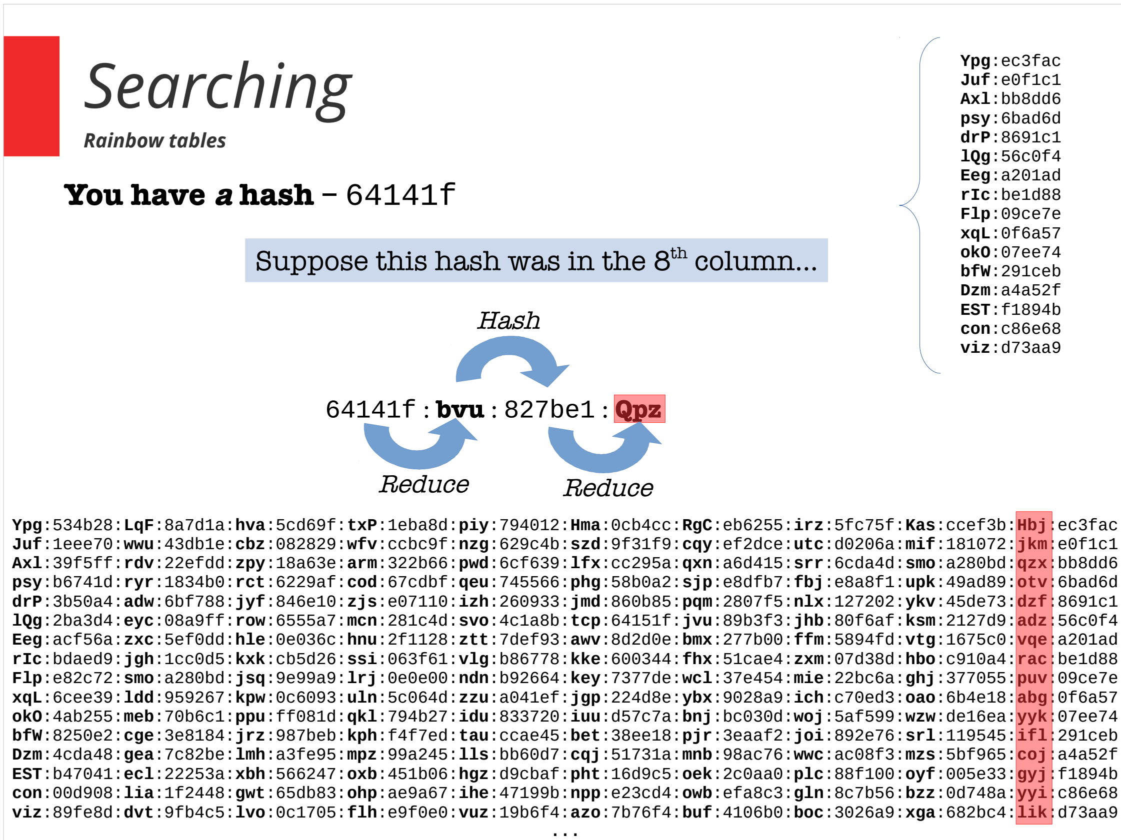
Reduce

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sjp:e8dfb7:fbj:e8a8f1:upk:49ad89:otv:6bad6d
drP:3b50a4:adw:6bf788:jyf:846e10:zjs:e07110:izh:260933:jmd:860b85:pqm:2807f5:nlx:127202:ykv:45de73:dzf:8691c1
lQg:2ba3d4:eyc:08a9ff:row:6555a7:mcn:281c4d:svo:4c1a8b:tcp:64151f:jvu:89b3f3:jhb:80f6af:ksm:2127d9:adz:56c0f4
Eeg:acf56a:zxc:5ef0dd:hle:0e036c:hnu:2f1128:ztt:7def93:awv:8d2d0e:bmx:277b00:ffm:5894fd:vtg:1675c0:vqe:a201ad
rIc:bdaed9:jgh:1cc0d5:kxk:cb5d26:ssi:063f61:vlg:b86778:kke:600344:fhx:51cae4:zxm:07d38d:hbo:c910a4:rac:be1d88
Flp:e82c72:smo:a280bd:jsq:9e99a9:lrj:0e0e00:ndn:b92664:key:7377de:wcl:37e454:mie:22bc6a:ghj:377055:puv:09ce7e
xqL:6cee39:ldd:959267:kpw:0c6093:uln:5c064d:zzu:a041ef:jgp:224d8e:ybx:9028a9:ich:c70ed3:oao:6b4e18:abg:0f6a57
okO:4ab255:meb:70b6c1:ppu:ff081d:qkl:794b27:idu:833720:iuu:d57c7a:bnj:bc030d:woj:5af599:wzw:de16ea:yyk:07ee74
bfW:8250e2:cge:3e8184:jrz:987beb:kph:f4f7ed:tau:ccae45:bet:38ee18:pjr:3eaaf2:joi:892e76:srl:119545:ifl:291ceb
Dzm:4cda48:gea:7c82be:lmh:a3fe95:mpz:99a245:lls:bb60d7:cqj:51731a:mnb:98ac76:wwc:ac08f3:mzs:5bf965:coj:a4a52f
EST:b47041:ecl:22253a:xbh:566247:oxb:451b06:hgz:d9cbaf:pht:16d9c5:oek:2c0aa0:plc:88f100:oyf:005e33:gyj:f1894b
con:00d908:lia:1f2448:gwt:65db83:ohp:ae9a67:ihe:47199b:npp:e23cd4:owb:efa8c3:gln:8c7b56:bzz:0d748a:yyi:c86e68
viz:89fe8d:dvt:9fb4c5:lvo:0c1705:flh:e9f0e0:vuz:19b6f4:azo:7b76f4:buf:4106b0:boc:3026a9:xga:682bc4:lik:d73aa9
...
```

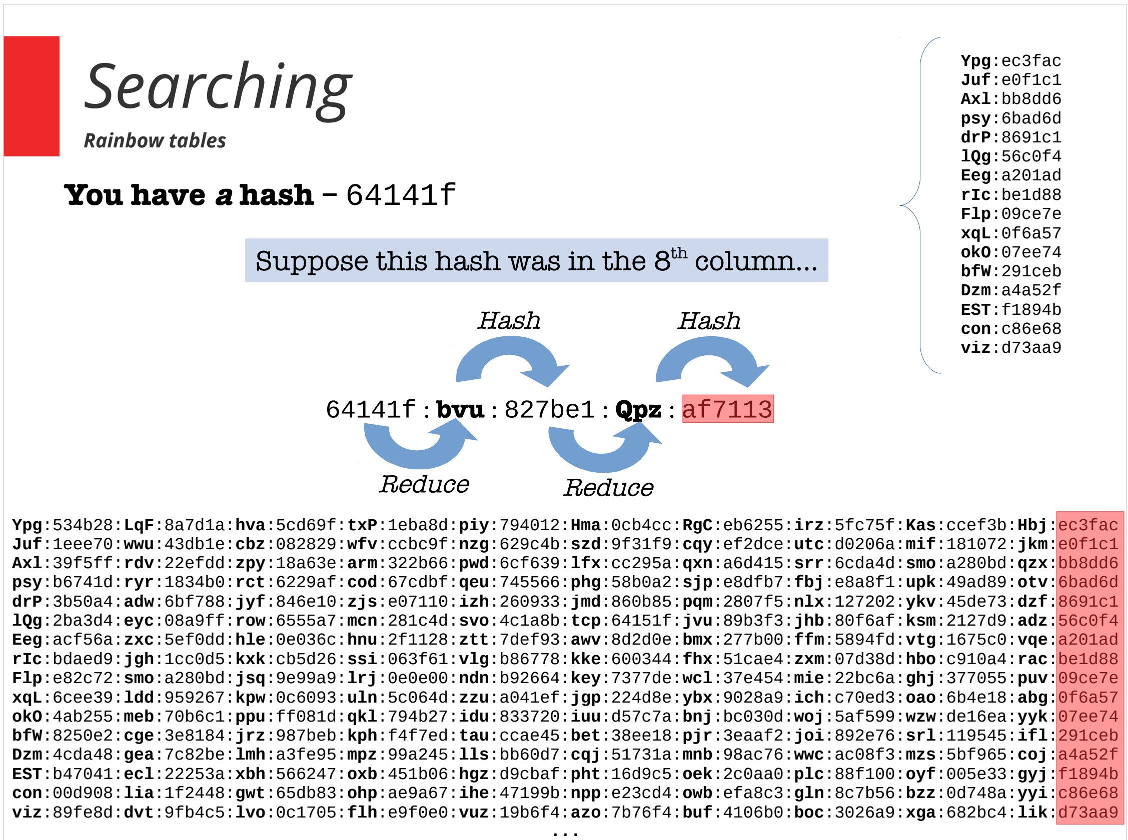
- Recalling the process used earlier to generate a chain by reducing a hash, and then hashing the key -
- Imagine this same reduce/hash process on our current hash.
- We first reduce our hash...



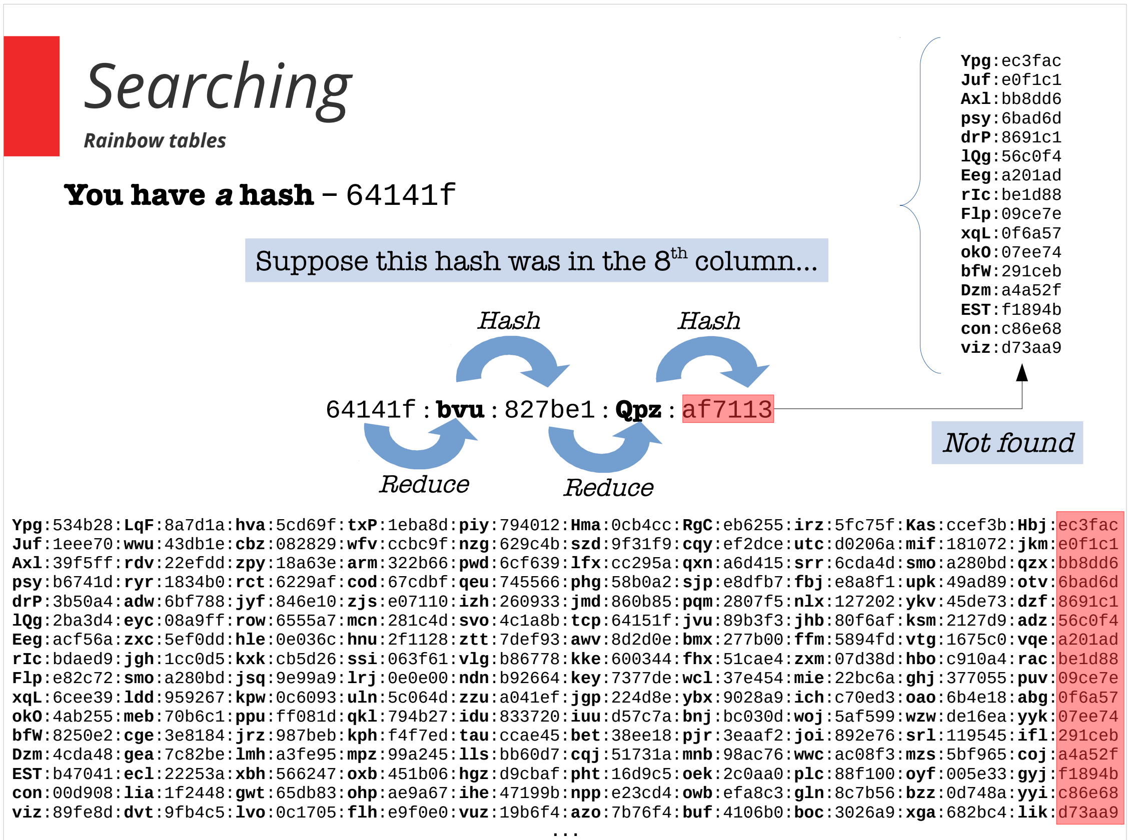
- … and then hash the resulting key.



- We reduce that new hash, and get a new key.

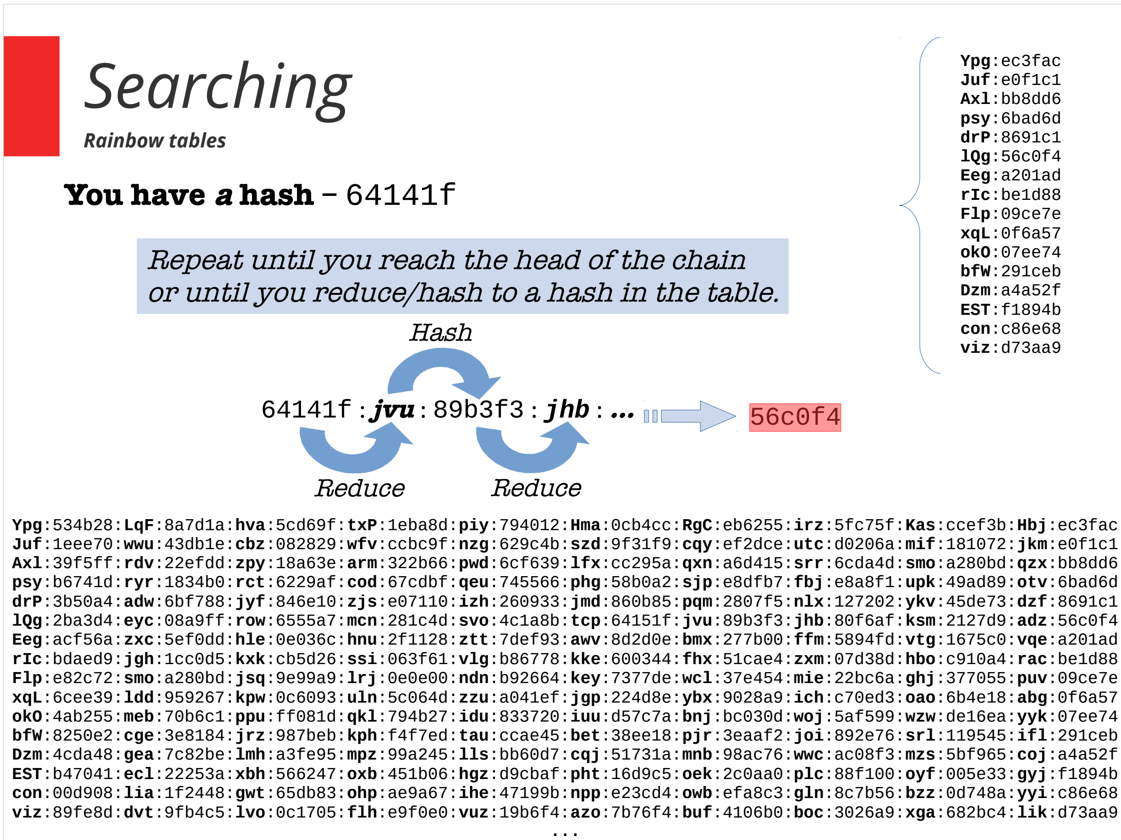


- Finally, we hash that key, getting a final hash.
- So *if* our hash existed in the 8th column of the larger table, through this process we should now have a hash that exists in the final column.

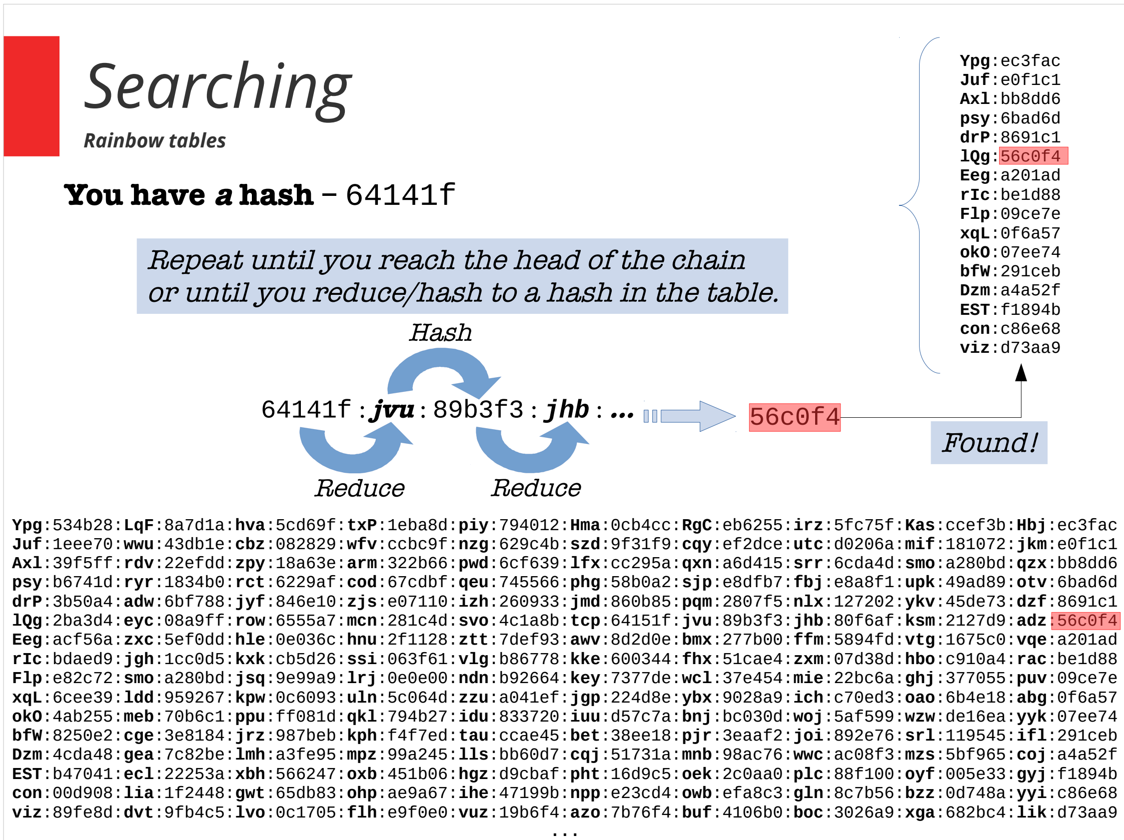

Searching
Rainbow tables
You have a hash - 64141f
Suppose this hash was in the 8th column...
Hash
Hash
64141f : bvu : 827be1 : Qpz : af7113
Reduce
Reduce
Not found
Ypg:ec3fac
Juf:e0f1c1
Axl:bb8dd6
psy:6bad6d
drP:8691c1
lQg:56c0f4
Eeg:a201ad
rIc:be1d88
Flp:09ce7e
xqL:0f6a57
okO:07ee74
bfW:291ceb
Dzm:a4a52f
EST:f1894b
con:c86e68
viz:d73aa9
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sjp:e8dfb7:fbj:e8a8f1:upk:49ad89:otv:6bad6d
drP:3b50a4:adw:6bf788:jyf:846e10:zjs:e07110:izh:260933:jmd:860b85:pqm:2807f5:nlx:127202:ykv:45de73:dzf:8691c1
lQg:2ba3d4:eyc:08a9ff:row:6555a7:mcn:281c4d:svo:4c1a8b:tcp:64151f:jvu:89b3f3:jhb:80f6af:ksm:2127d9:adz:56c0f4
Eeg:acf56a:zxc:5ef0dd:hle:0e036c:hnu:2f1128:ztt:7def93:awv:8d2d0e:bmx:277b00:ffm:5894fd:vtg:1675c0:vqe:a201ad
rIc:bdaed9:jgh:1cc0d5:kxk:cb5d26:ssi:063f61:vlg:b86778:kke:600344:fhx:51cae4:zxm:07d38d:hbo:c910a4:rac:be1d88
Flp:e82c72:smo:a280bd:jsq:9e99a9:lrj:0e0e00:ndn:b92664:key:7377de:wcl:37e454:mie:22bc6a:ghj:377055:puv:09ce7e
xqL:6cee39:ldd:959267:kpw:0c6093:uln:5c064d:zzu:a041ef:jgp:224d8e:ybx:9028a9:ich:c70ed3:oao:6b4e18:abg:0f6a57
okO:4ab255:meb:70b6c1:ppu:ff081d:qkl:794b27:idu:833720:iuu:d57c7a:bnj:bc030d:woj:5af599:wzw:de16ea:yyk:07ee74
bfW:8250e2:cge:3e8184:jrz:987beb:kph:f4f7ed:tau:ccae45:bet:38ee18:pjr:3eaaf2:joi:892e76:srl:119545:ifl:291ceb
Dzm:4cda48:gea:7c82be:lmh:a3fe95:mpz:99a245:lls:bb60d7:cqj:51731a:mnb:98ac76:wwc:ac08f3:mzs:5bf965:coj:a4a52f
EST:b47041:ecl:22253a:xbh:566247:oxb:451b06:hgz:d9cbaf:pht:16d9c5:oek:2c0aa0:plc:88f100:oyf:005e33:gyj:f1894b
con:00d908:lia:1f2448:gwt:65db83:ohp:ae9a67:ihe:47199b:npp:e23cd4:owb:efa8c3:gln:8c7b56:bzz:0d748a:yyi:c86e68
viz:89fe8d:dvt:9fb4c5:lvo:0c1705:flh:e9f0e0:vuz:19b6f4:azo:7b76f4:buf:4106b0:boc:3026a9:xga:682bc4:lik:d73aa9
...

- But we don't.

X



- This process is repeated, picking a column the hash you hold *might* have been in, and running it down a reduce/hash chain to see if the resulting hash exists in the reduced table.



- You'll either reach the front of the table, not finding a match,
- Or you'll find a chain location that results in a hash that can be found in the reduced table.

X

Searching

Rainbow tables

You have *a* hash – 64141f

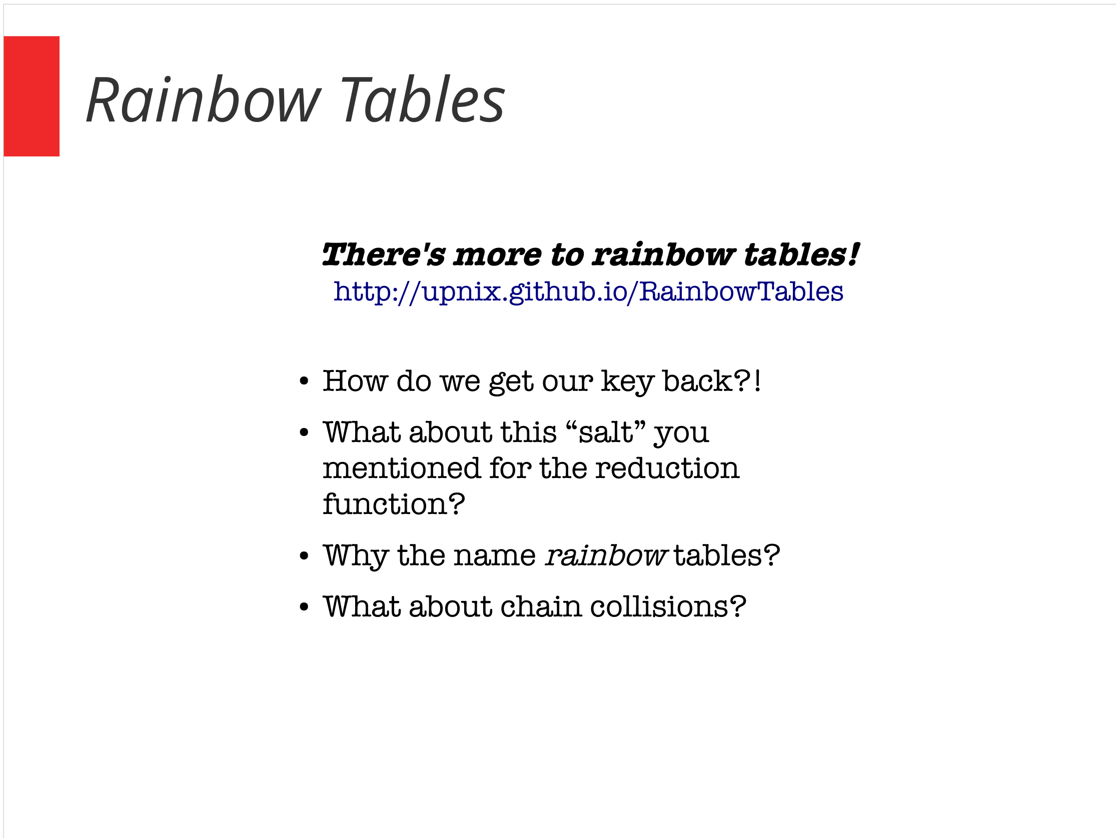
Our hash found after 4 reduce/hashes

```
Ypg:ec3fac
Juf:e0f1c1
Axl:bb8dd6
psy:6bad6d
drP:8691c1
lQg:56c0f4
Eeg:a201ad
rIc:be1d88
Flp:09ce7e
xqL:0f6a57
okO:07ee74
bfW:291ceb
Dzm:a4a52f
EST:f1894b
con:c86e68
viz:d73aa9
```

64141f : ***jvu*** : 89b3f3 : ***jhb*** : ***...*** → 56c0f4

```
Ypg:534b28:LqF:8a7d1a:hva:5cd69f:txP:1eba8d:piy:794012:Hma:0cb4cc:RgC:eb6255:irz:5fc75f:Kas:ccef3b:Hbj:ec3fac
Juf:1eee70:wwu:43db1e:cbz:082829:wfv:ccbc9f:nzg:629c4b:szd:9f31f9:cqy:ef2dce:utc:d0206a:mif:181072:jkm:e0f1c1
Axl:39f5ff:rdv:22efdd:zpy:18a63e:arm:322b66:pwd:6cf639:lfx:cc295a:qxn:a6d415:srr:6cda4d:smo:a280bd:qzx:bb8dd6
psy:b6741d:ryr:1834b0:rct:6229af:cod:67cdbf:qeu:745566:phg:58b0a2:sj[illegible]dfb7:fbj[illegible]8f1:up[illegible]d89:ot[illegible]d6d
drP:3b50a4:adw:6bf788:jyf:846e10:zjs:e07110:izh:260933:jmd:860b85:p[illegible]:28[illegible]f5:nl[illegible]:12[illegible]02:y[illegible]:45[illegible]73:d[illegible]:86[illegible]c1
lQg:2ba3d4:eyc:08a9ff:row:6555a7:mcn:281c4d:svo:4c1a8b:tcp:64151f:jvu:89b3f3:jhb:80f6af:ksm:2127d9:adz:56c0f4
Eeg:acf56a:zxc:5ef0dd:hle:0e036c:hnu:2f1128:ztt:7def93:awv:8d2[illegible]0e:[illegible]x:277b00:f[illegible]m:5894fd:[illegible]q:167[illegible]c0:[illegible]e:a201ad
rIc:bdaed9:jgh:1cc0d5:kxk:cb5d26:ssi:063f61:vlg:b86778:kke:600[illegible]:[illegible]nx:51ca[illegible]:[illegible]m:07d3[illegible]:[illegible]o:c91[illegible]:[illegible]ac:be1d88
Flp:e82c72:smo:a280bd:jsq:9e99a9:lrj:0e0e00:ndn:b92664:key:7377de:wcl:37e454:mie:22bc6a:ghj:377055:puv:09ce7e
xqL:6cee39:ldd:959267:kpw:0c6093:uln:5c064d:zzu:a041ef:jgp:224d8e:ybx:9028a9:ich:c70ed3:oao:6b4e18:abg:0f6a57
okO:4ab255:meb:70b6c1:ppu:ff081d:qkl:794b27:idu:833720:iuu:d57c7a:bnj:bc030d:woj:5af599:wzw:de16ea:yyk:07ee74
bfW:8250e2:cge:3e8184:jrz:987beb:kph:f4f7ed:tau:ccae45:bet:38ee18:pjr:3eaaf2:joi:892e76:srl:119545:ifl:291ceb
Dzm:4cda48:gea:7c82be:lmh:a3fe95:mpz:99a245:lls:bb60d7:cqj:51731a:mnb:98ac76:wwc:ac08f3:mzs:5bf965:coj:a4a52f
EST:b47041:ecl:22253a:xbh:566247:oxb:451b06:hgz:d9cbaf:pht:16d9c5:oek:2c0aa0:plc:88f100:oyf:005e33:gyj:f1894b
con:00d908:lia:1f2448:gwt:65db83:ohp:ae9a67:ihe:47199b:npp:e23cd4:owb:efa8c3:gln:8c7b56:bzz:0d748a:yyi:c86e68
viz:89fe8d:dvt:9fb4c5:lvo:0c1705:flh:e9f0e0:vuz:19b6f4:azo:7b76f4:buf:4106b0:boc:3026a9:xga:682bc4:lik:d73aa9
...
```

- For our example hash, when 4 reduce/hash iterations are applied, we end up with a hash that is in the reduced table.



- And that's a very good thing...