Master’s thesis

IOC sharing and sightings

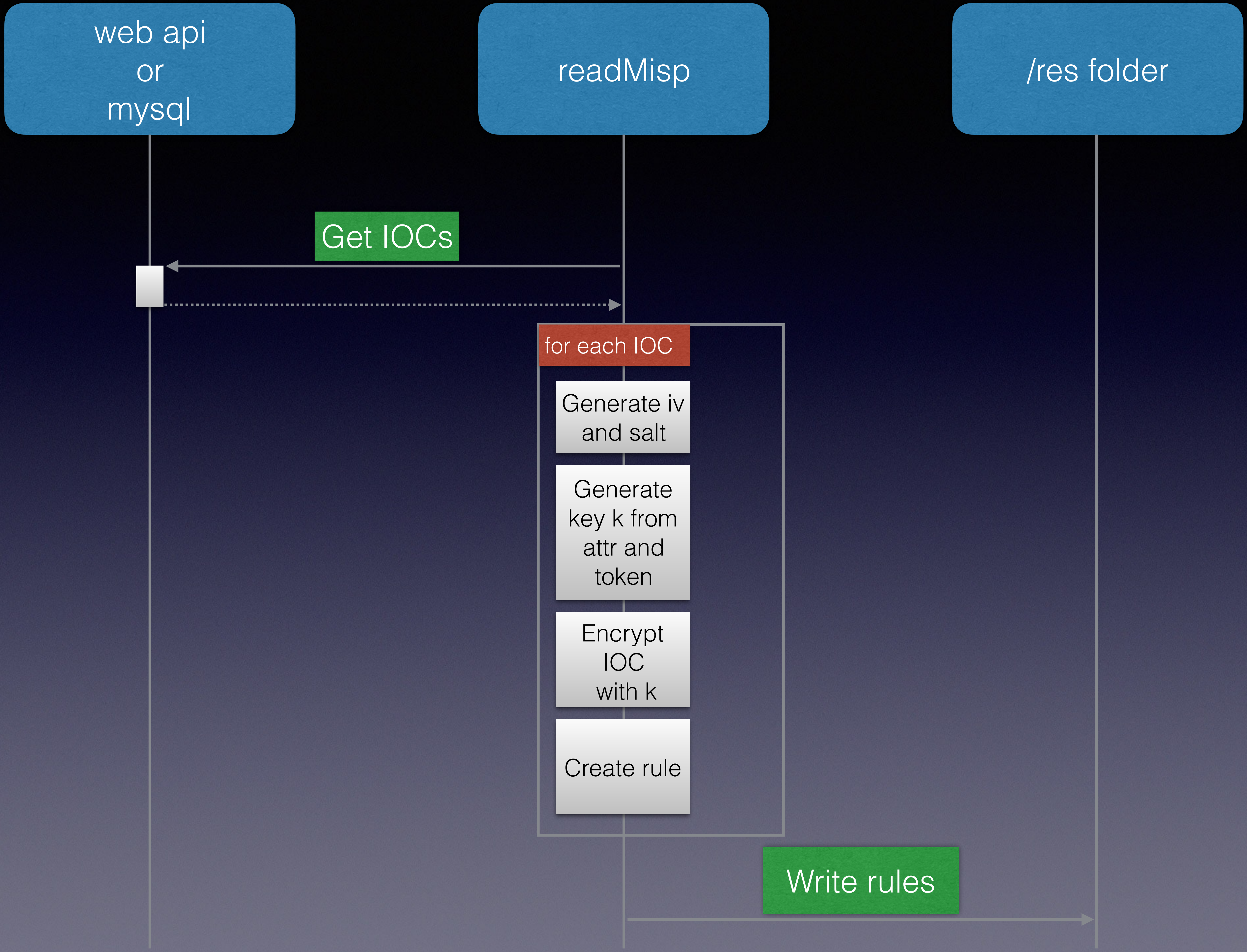
Paper implemented

van de Kamp, T., Peter, A., Everts, M. H., & Jonker, W. (2016, October). Private Sharing of IOCs and Sightings. In Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security (pp. 35-38). ACM.

Sharing

Read and Transform rules

Check value for IOC matches



web api
or
mysql

readMisp

/res folder

Get IOCs

for each IOC

Generate iv and salt

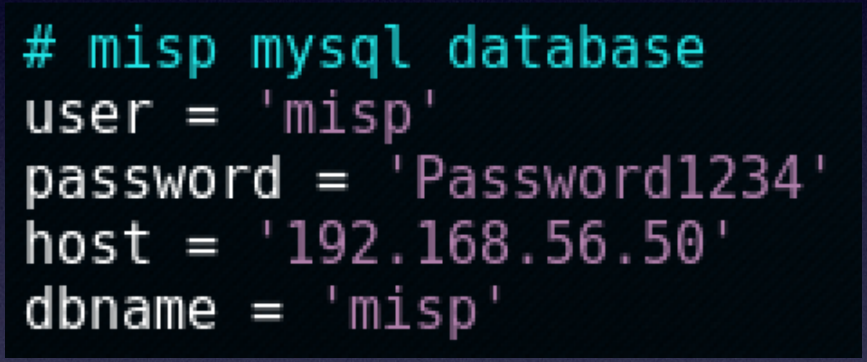
Generate key k from attr and token

Encrypt IOC with k

Create rule

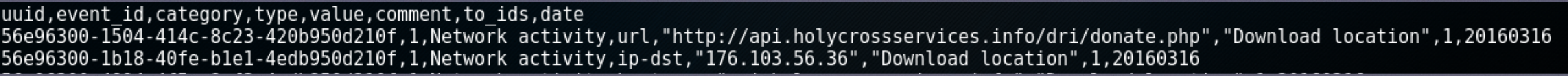
Write rules

mysql or web api

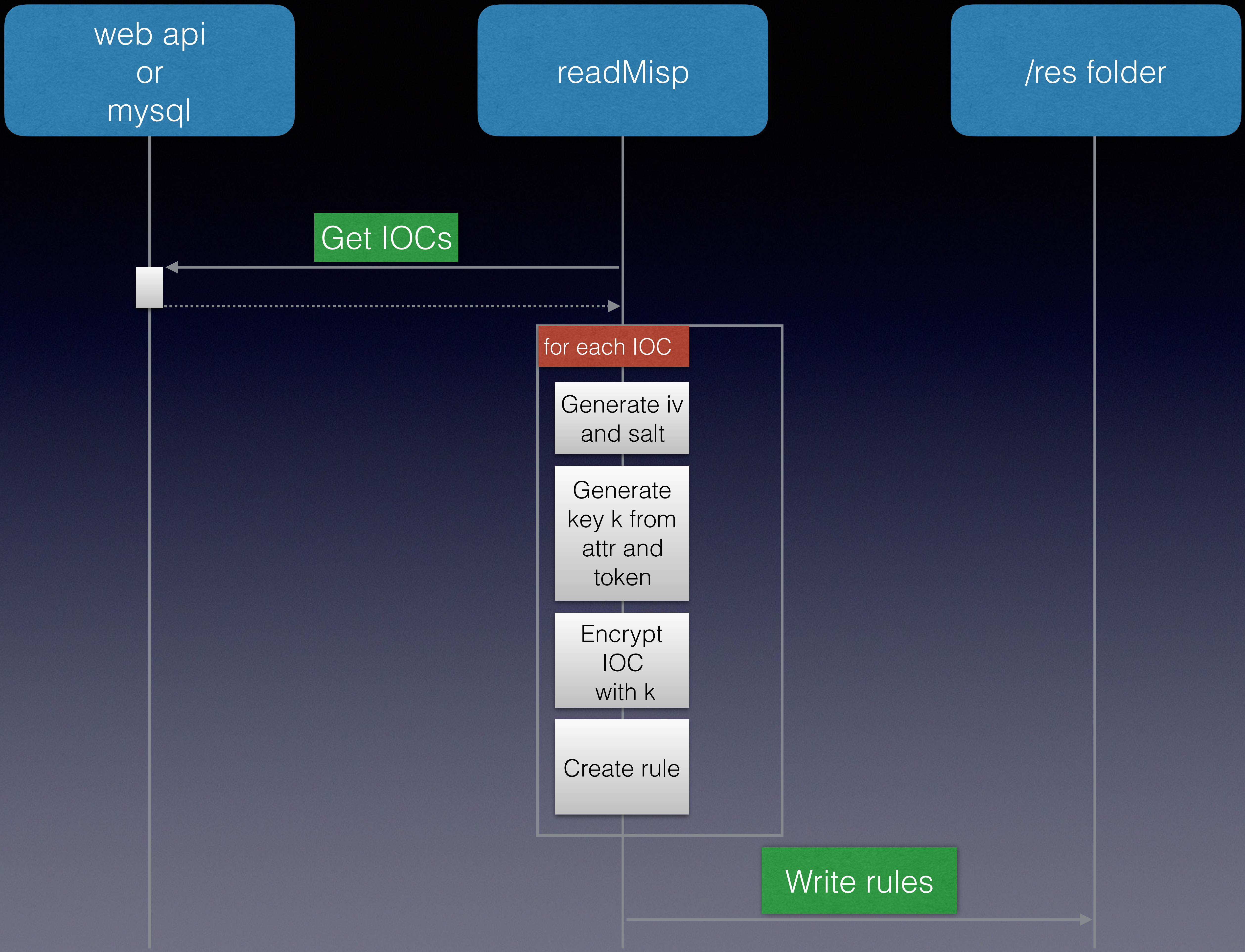


```
# misp mysql database
user = 'misp'
password = 'Password1234'
host = '192.168.56.50'
dbname = 'misp'
```

./update.py (Automatically started from readMisp)



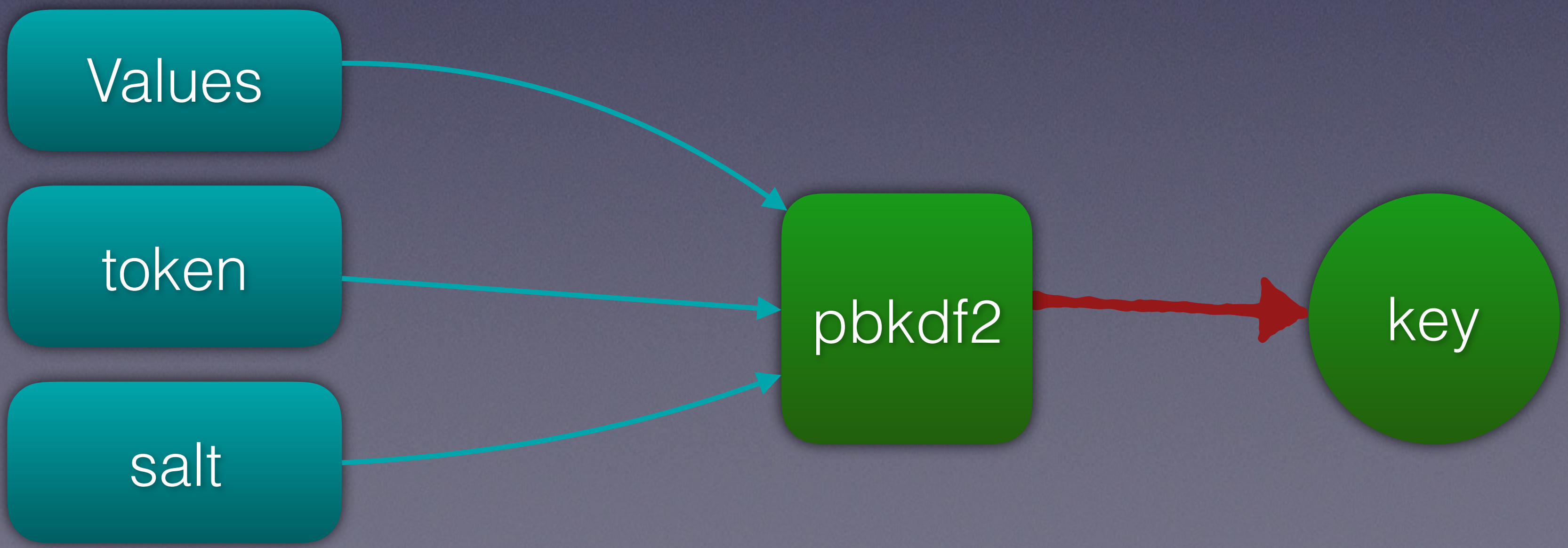
```
uuid,event_id,category,type,value,comment,to_ids,date
56e96300-1504-414c-8c23-420b950d210f,1,Network activity,url,"http://api.holycrossservices.info/dri/donate.php","Download location",1,20160316
56e96300-1b18-40fe-b1e1-4edb950d210f,1,Network activity,ip-dst,"176.103.56.36","Download location",1,20160316
```


web api
or
mysql
readMisp
/res folder
Get IOCs
for each IOC
Generate iv and salt
Generate key k from attr and token
Encrypt IOC with k
Create rule
Write rules

Pbkdf2 vs HKDF

Slow down brute force thanks to iterations

Designed to be « random looking » directly



Values

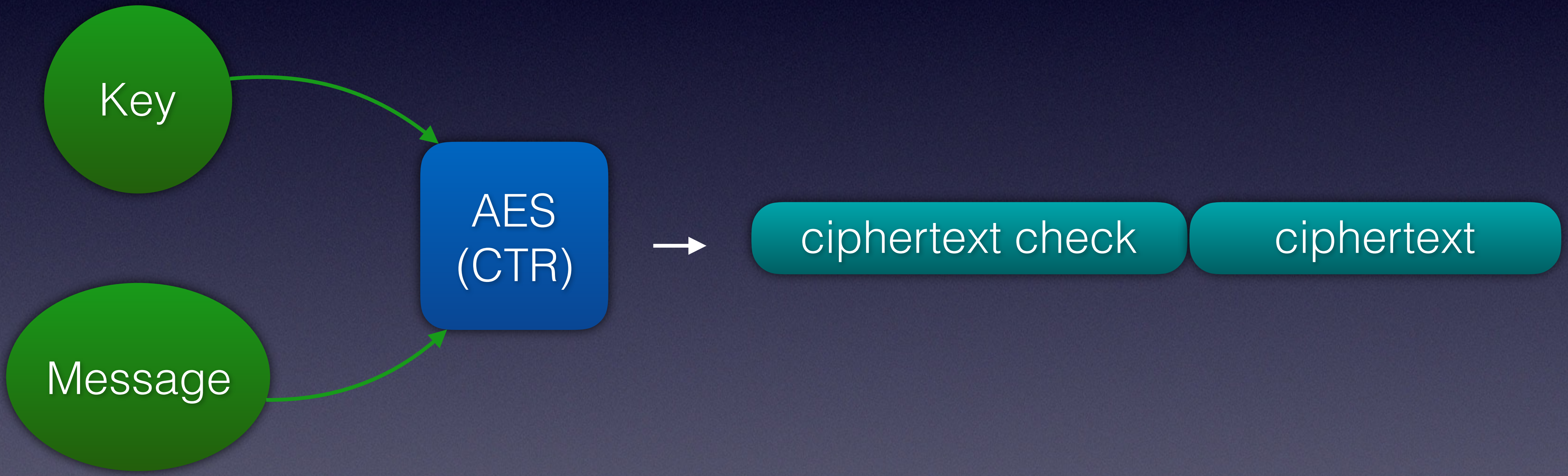
token

salt

pbkdf2

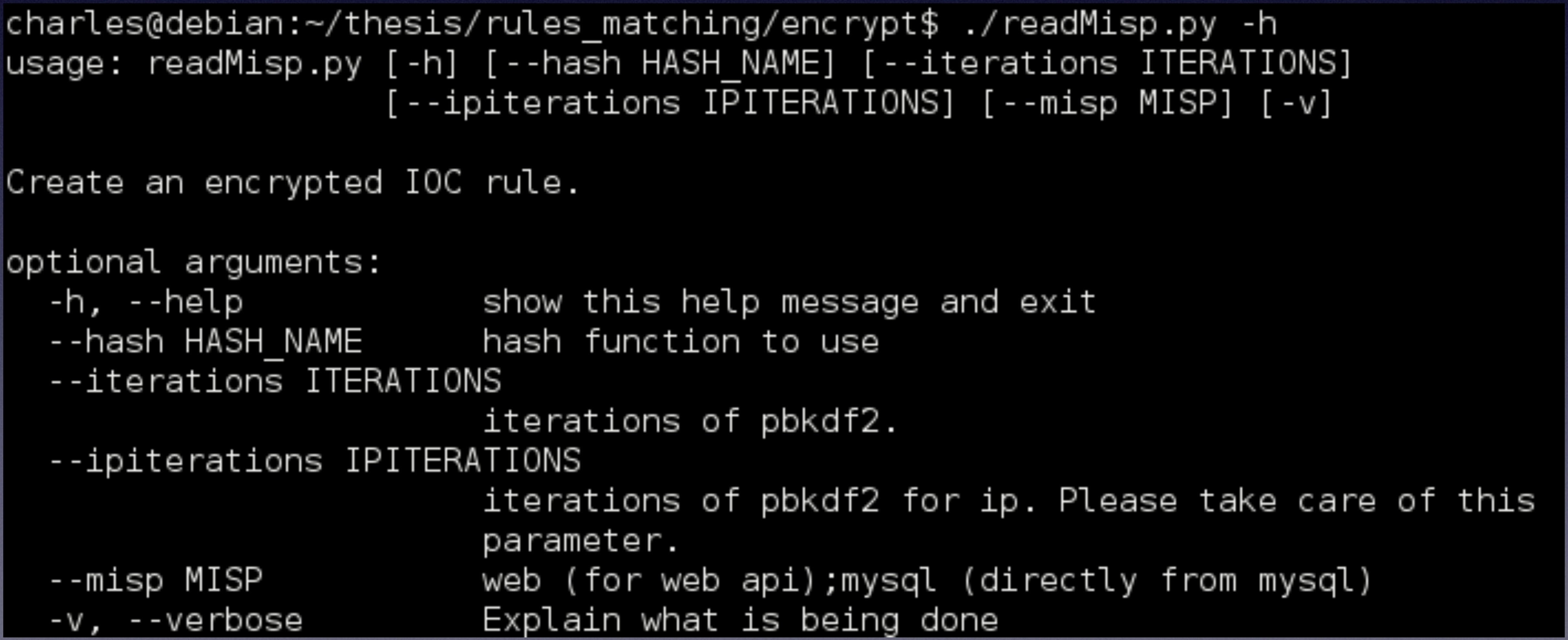
key

Encryption



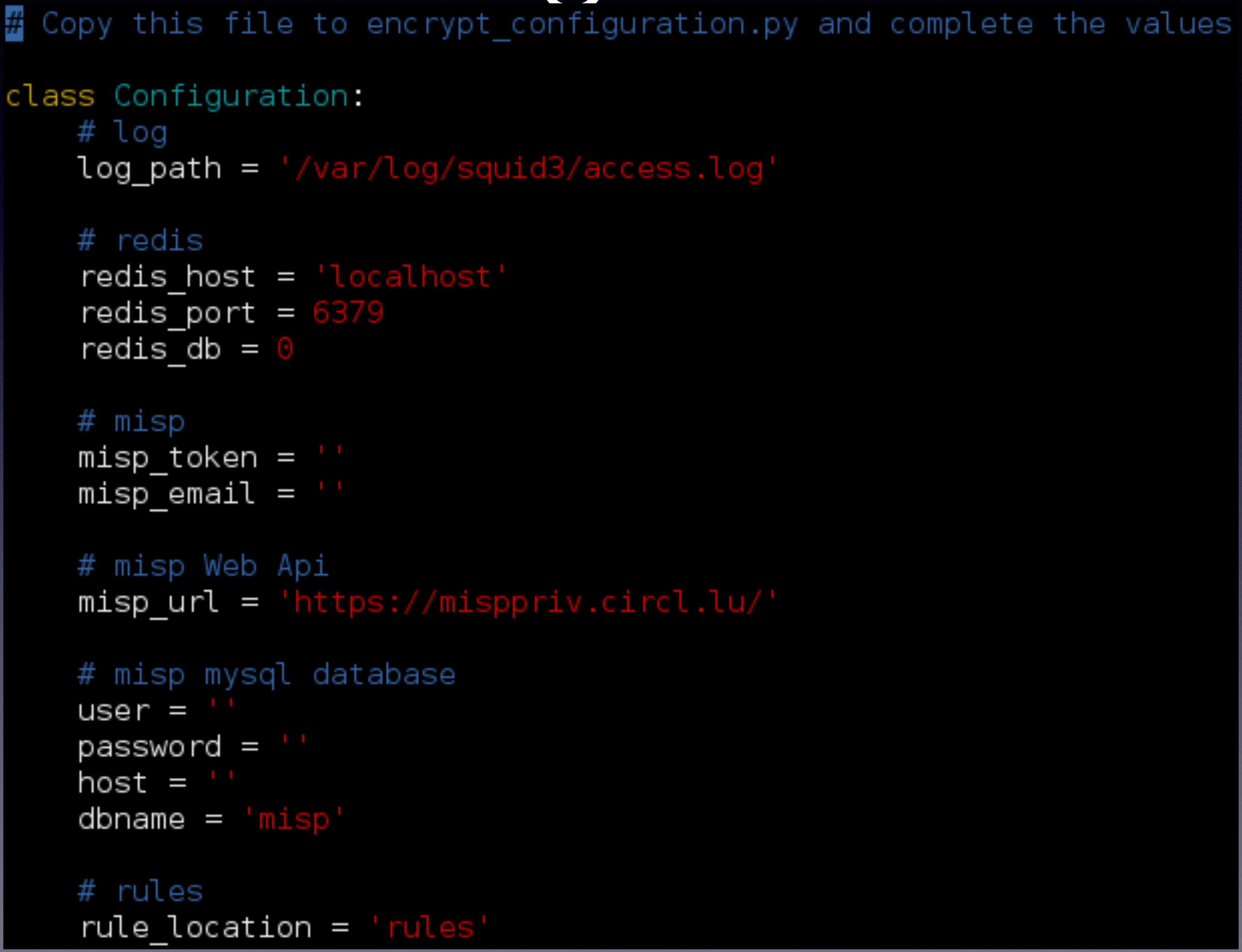
Read and Transform

./readMisp.py -h



```
charles@debian:~/thesis/rules_matching/encrypt$ ./readMisp.py -h
usage: readMisp.py [-h] [--hash HASH_NAME] [--iterations ITERATIONS]
                   [--ipiterations IPITERATIONS] [--misp MISP] [-v]

Create an encrypted IOC rule.

optional arguments:
  -h, --help            show this help message and exit
  --hash HASH_NAME      hash function to use
  --iterations ITERATIONS
                        iterations of pbkdf2.
  --ipiterations IPITERATIONS
                        iterations of pbkdf2 for ip. Please take care of this
                        parameter.
  --misp MISP           web (for web api);mysql (directly from mysql)
  -v, --verbose         Explain what is being done
```

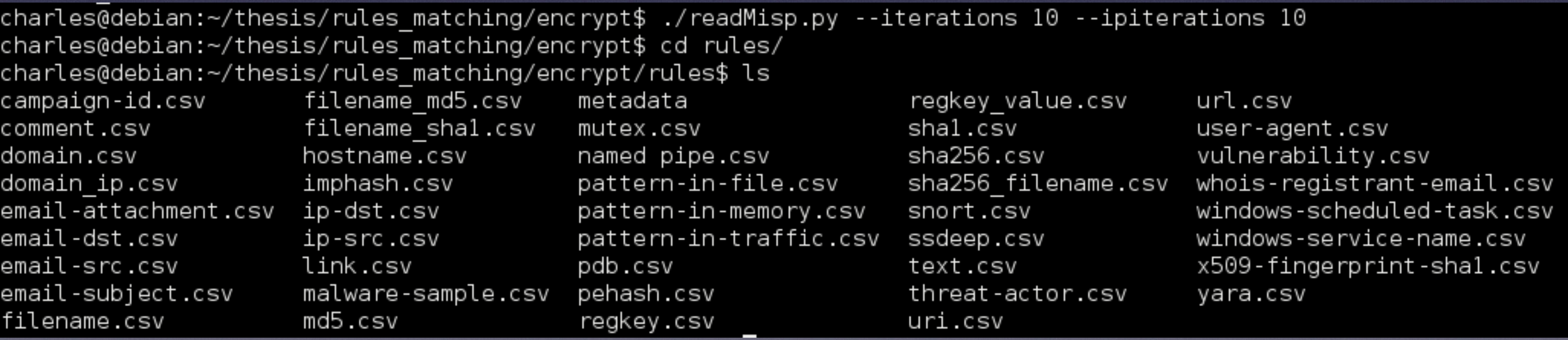
Configuration



```
# Copy this file to encrypt_configuration.py and complete the values

class Configuration:
    # log
    log_path = '/var/log/squid3/access.log'

    # redis
    redis_host = 'localhost'
    redis_port = 6379
    redis_db = 0

    # misp
    misp_token = ''
    misp_email = ''

    # misp Web Api
    misp_url = 'https://misppriv.circl.lu/'

    # misp mysql database
    user = ''
    password = ''
    host = ''
    dbname = 'misp'

    # rules
    rule_location = 'rules'
```

Rules

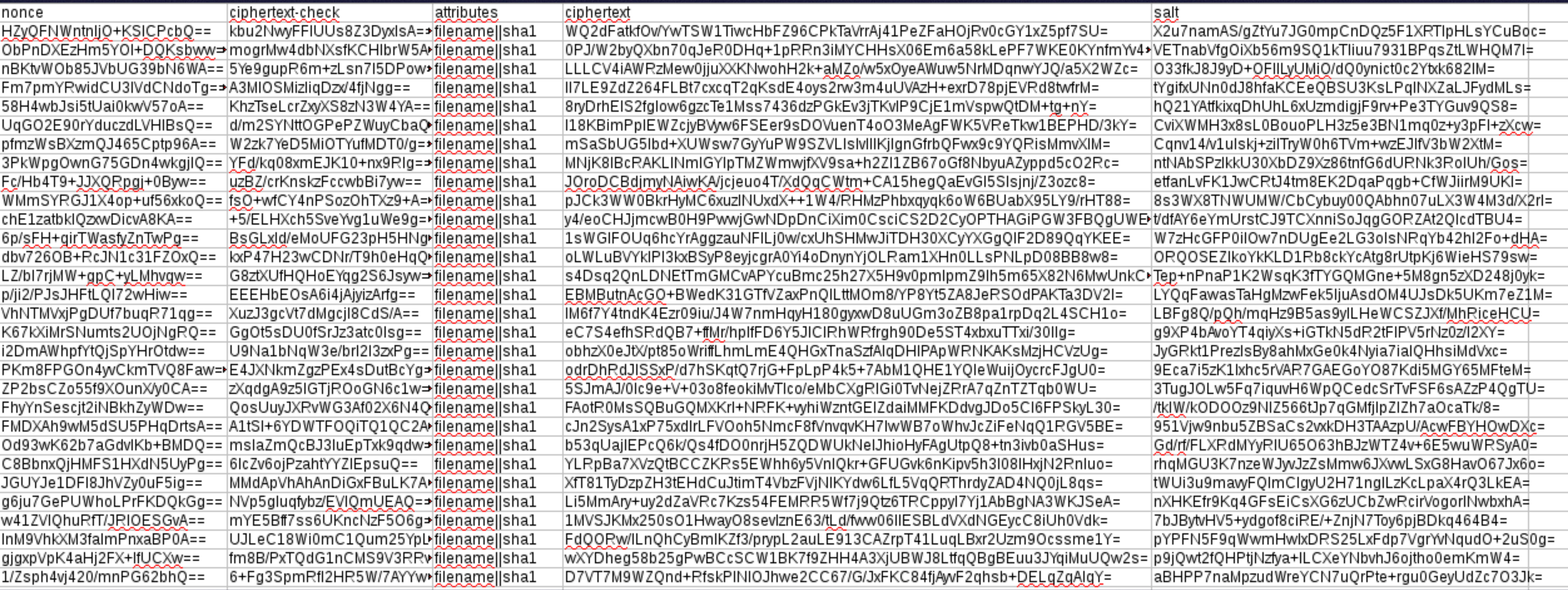
Rules are stored in files according to the type of IOC



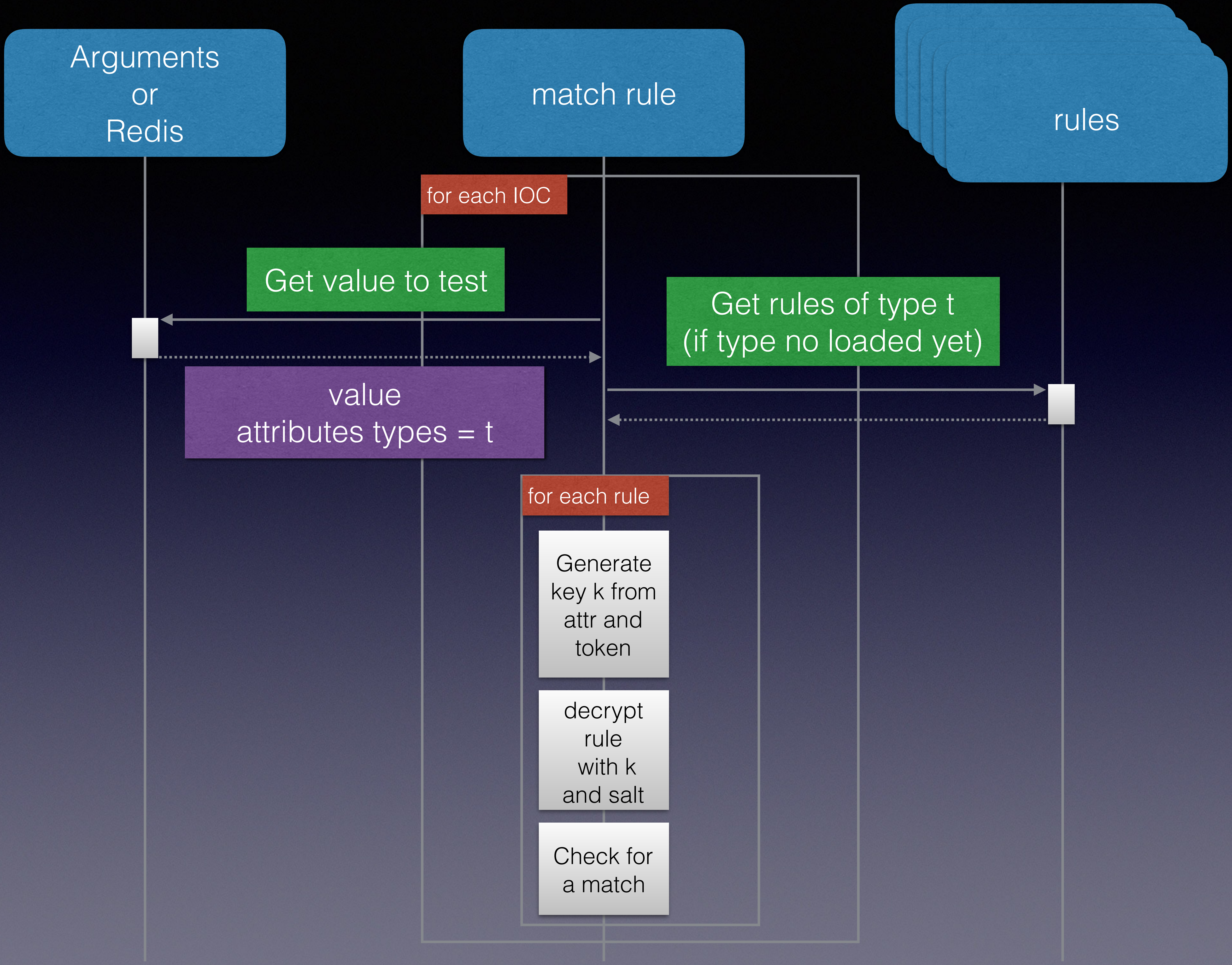
```
charles@debian:~/thesis/rules_matching/encrypt$ ./readMisp.py --iterations 10 --ipiterations 10
charles@debian:~/thesis/rules_matching/encrypt$ cd rules/
charles@debian:~/thesis/rules_matching/encrypt/rules$ ls
campaign-id.csv       filename_md5.csv    metadata                regkey_value.csv     url.csv
comment.csv           filename_sha1.csv   mutex.csv               sha1.csv             user-agent.csv
domain.csv            hostname.csv        named pipe.csv          sha256.csv           vulnerability.csv
domain_ip.csv         imphash.csv         pattern-in-file.csv     sha256_filename.csv  whois-registrant-email.csv
email-attachment.csv  ip-dst.csv          pattern-in-memory.csv   snort.csv            windows-scheduled-task.csv
email-dst.csv         ip-src.csv          pattern-in-traffic.csv  ssdeep.csv           windows-service-name.csv
email-src.csv         link.csv            pdb.csv                 text.csv             x509-fingerprint-sha1.csv
email-subject.csv     malware-sample.csv  pehash.csv              threat-actor.csv     yara.csv
filename.csv          md5.csv             regkey.csv              uri.csv
```

Rules

MISP token is also used to generate rules



nonce	ciphertext-check	attributes	ciphertext	salt
HZyQFNWntnIjO+KSICPcbQ==	kbu2NwyFFIUUs8Z3DyxIsA=	filename||sha1	WQ2dFatkfOv/YwTSW1TiwcHbFZ96CPkTaVrrAj41PeZFaHOjRv0cGY1xZ5pf7SU=	X2u7namAS/gZtYu7JG0mpCnDQz5F1XRTIpHLsYCuBoc=
ObPnDXEzHm5YOI+DQKsbww=	mogrMw4dbNXsfKCHIbrW5A	filename||sha1	0PJ/W2byQXbn70qJeR0DHq+1pRRn3iMYCHHsX06Em6a58kLePF7WKE0KYnfmYv4	VETnabVfgOiXb56m9SQ1kTliuu7931BPqsZtLWHQM7I=
nBKtvWOb85JVbUG39bN6WA==	5Ye9gupR6m+zLsn7I5DPow	filename||sha1	LLLCV4iAWRzMew0jjuXXKNwohH2k+aMZo/w5xOyeAWuw5NrMDqnwYJQ/a5X2WZc=	O33fkJ8J9yD+OFIILyUMiO/dQ0ynict0c2Ytxk682IM=
Fm7pmYRwidCU3IVdCNdoTg=	A3MIOSMizliqDzx/4fjNgg==	filename||sha1	lI7LE9ZdZ264FLBt7cxcqT2qKsdE4oys2rw3m4uUVAzH+exrD78pjEVRd8twfrM=	tYgifxUNn0dJ8hfaKCEeQBSU3KsLPqINXZaLJFydMLs=
58H4wbJsi5tUai0kwV57oA==	KhzTseLcrZxyXS8zN3W4YA==	filename||sha1	8ryDrhEIS2fglow6gzcTe1Mss7436dzPGkEv3jTKvIP9CjE1mVspwQtDM+tg+nY=	hQ21YAtfkixqDhUhL6xUzmdigjF9rv+Pe3TYGuv9QS8=
UqGO2E90rYduczdLVHIBsQ==	d/m2SYNttOGPePZWuyCbaQ	filename||sha1	I18KBimPpIEWZcjyBVyw6FSEer9sDOVuenT4oO3MeAgFWK5VReTkw1BEPHD/3kY=	CviXWMH3x8sL0BouoPLH3z5e3BN1mq0z+y3pFI+zXcw=
pfmzWsBXzmQJ465Cptp96A==	W2zk7YeD5MiOTYufMDT0/g=	filename||sha1	mSaSbUG5Ibd+XUWsw7GyYuPW9SZVLIsIvIIIKjIgnGfrbQFwx9c9YQRisMmvXIM=	Cqnv14/v1uIskj+ziITryW0h6TVm+wzEJIfV3bW2XtM=
3PkWpgOwnG75GDn4wkgjIQ==	YFd/kq08xmEJK10+nx9RIg=	filename||sha1	MNjK8IBcRAKLINmIGYIpTMZWmwjfXV9sa+h2ZI1ZB67oGf8NbyuAZyppd5cO2Rc=	ntNAbSPzIkkU30XbDZ9Xz86tnfG6dURNk3RoIUh/Gos=
Fc/Hb4T9+JJXQRpgj+0Byw==	uzBZ/crKnskzFccwbBi7yw==	filename||sha1	JOroDCBdjmyNAiwKA/jcjeuo4T/XdQqCWtm+CA15hegQaEvGI5SIsjnj/Z3ozc8=	etfanLvFK1JwCRtJ4tm8EK2DqaPqgb+CfWJiirM9UKI=
WMmSYRGJ1X4op+uf56xkoQ==	fsO+wfCY4nPSozOhTXz9+A=	filename||sha1	pJCk3WW0BkrHyMC6xuzINUxdX++1W4/RHMzPhbxqyqk6oW6BUabX95LY9/rHT88=	8s3WX8TNWUMW/CbCybuy00QAbhn07uLX3W4M3d/X2rI=
chE1zatbkIQzxwDicvA8KA==	+5/ELHXch5SveYvg1uWe9g=	filename||sha1	y4/eoCHJjmcwB0H9PwwjGwNDpDnCiXim0CsciCS2D2CyOPTHAGiPGW3FBQgUWE	t/dfAY6eYmUrstCJ9TCXnniSoJqgGORZAt2QIcdTBU4=
6p/sFH+qirTWasfyZnTwPg==	BsGLxId/eMoUFG23pH5HNg	filename||sha1	1sWGIFOUq6hcYrAggzauNFILj0w/cxUhSHMwJiTDH30XCyYXGgQIF2D89QqYKEE=	W7zHcGFP0iIOw7nDUgEe2LG3oIsNRqYb42hI2Fo+dHA=
dbv726OB+RcJN1c31FZOxQ==	kxP47H23wCDNr/T9h0eHqQ	filename||sha1	oLWLuBVYkIPI3kxBSyP8eyjcgrA0Yi4oDnynYjOLRam1XHn0LLsPNLpD08BB8w8=	ORQOSEZIkoYkKLD1Rb8ckYcAtg8rUtpKj6WieHS79sw=
LZ/bI7rjMW+gpC+yLMhvqw==	G8ztXUfHQHoEYqg2S6Jsyw=	filename||sha1	s4Dsq2QnLDNEtTmGMCvAPYcuBmc25h27X5H9v0pmIpmZ9Ih5m65X82N6MwUnkC	Tep+nPnaP1K2WsqK3fTYGQMGne+5M8gn5zXD248j0yk=
p/ji2/PJsJHFtLQI72wHiw==	EEEHbEOsA6i4jAjyizArfg==	filename||sha1	EBMButnAcGO+BWedK31GTfVZaxPnQILttMOm8/YP8Yt5ZA8JeRSOdPAKTa3DV2I=	LYQqFawasTaHgMzwFek5IjuAsdOM4UJsDk5UKm7eZ1M=
VhNTMVxjPgDUf7buqR71qg==	XuzJ3gcVt7dMgcjI8CdS/A==	filename||sha1	IM6f7Y4tndK4Ezr09iu/J4W7nmHqyH180gyxwD8uUGm3oZB8pa1rpDq2L4SCH1o=	LBFg8Q/pQh/mqHz9B5as9yILHeWCSZJXf/MhRiceHCU=
K67kXiMrSNumts2UOjNgRQ==	GgOt5sDU0fSrJz3atc0Isg==	filename||sha1	eC7S4efhSRdQB7+ffMr/hpIfFD6Y5JICIRhWRfrgh90De5ST4xbxuTTxi/30IIg=	g9XP4bAvoYT4qiyXs+iGTkN5dR2tFIPV5rNz0z/I2XY=
i2DmAWhpfYtQjSpYHrOtdw==	U9Na1bNqW3e/brI2I3zxPg==	filename||sha1	obhzX0eJtX/pt85oWriffLhmLmE4QHGxTnaSzfAIqDHIPApWRNKAKsMzjHCVzUg=	JyGRkt1PrezIsBy8ahMxGe0k4Nyia7iaIQHhsiMdVxc=
PKm8FPGOn4ywCkmTVQ8Faw=	E4JXNkmZgzPEx4sDutBcYg=	filename||sha1	odrDhRdJISSxP/d7hSKqtQ7rjG+FpLpP4k5+7AbM1QHE1YQIeWuijOycrcFJgU0=	9Eca7i5zK1Ixhc5rVAR7GAEGoYO87Kdi5MGY65MFteM=
ZP2bsCZo55f9XOunX/y0CA==	zXqdgA9z5IGTjROoGN6c1w=	filename||sha1	5SJmAJ/0Ic9e+V+03o8feokiMvTIco/eMbCXgRIGi0TvNejZRrA7qZnTZTqb0WU=	3TugJOLw5Fq7iquvH6WpQCedcSrTvFSF6sAZzP4QgTU=
FhyYnSescjt2iNBkhZyWDw==	QosUuyJXRvWG3Af02X6N4Q	filename||sha1	FAotR0MsSQBuGQMXKrI+NRFK+vyhiWzntGEIZdaiMMFKDdvgJDo5CI6FPSkyL30=	/tkIW/kODOOz9NIZ566tJp7qGMfjIpZIZh7aOcaTk/8=
FMDXAh9wM5dSU5PHqDrtsA==	A1tSI+6YDWTFOQiTQ1QC2A	filename||sha1	cJn2SysA1xP75xdIrLFVOoh5NmcF8fVnvqvKH7IwWB7oWhvJcZiFeNqQ1RGV5BE=	951Vjw9nbu5ZBSaCs2vxkDH3TAAzpU/AcwFBYHOwDXc=
Od93wK62b7aGdvIKb+BMDQ==	msIaZmQcBJ3IuEpTxk9qdw=	filename||sha1	b53qUajIEPcQ6k/Qs4fDO0nrjH5ZQDWUkNeIJhioHyFAgUtpQ8+tn3ivb0aSHus=	Gd/rf/FLXRdMYyRIU65O63hBJzWTZ4v+6E5wuWRSyA0=
C8BbnxQjHMFS1HXdN5UyPg==	6IcZv6ojPzahtYYZIEpsuQ==	filename||sha1	YLRpBa7XVzQtBCCZKRs5EWhh6y5VnIQkr+GFUGvk6nKipv5h3I08IHxjN2RnIuo=	rhqMGU3K7nzeWJywJzZsMmw6JXwwLSxG8HavO67Jx6o=
JGUYJe1DFI8JhVZy0uF5ig==	MMdApVhAhAnDiGxFBuLK7A	filename||sha1	XfT81TyDzpZH3tEHdCuJtimT4VbzFVjNIKYdw6LfL5VqQRThrdyZAD4NQ0jL8qs=	tWUi3u9mavyFQImCIgyU2H71ngILzKcLpaX4rQ3LkEA=
g6ju7GePUWhoLPrFKDQkGg==	NVp5gluqfybz/EVIQmUEAQ=	filename||sha1	Li5MmAry+uy2dZaVRc7Kzs54FEMRR5Wf7j9Qtz6TRCppyI7Yj1AbBgNA3WKJSeA=	nXHKEfr9Kq4GFsEiCsXG6zUCbZwRcirVogorINwbxhA=
w41ZVIQhuRfT/JRIOESGvA==	mYE5Bff7ss6UKncNzF5O6g	filename||sha1	1MVSJKMx250sO1HwayO8sevIznE63/tLd/fww06IIESBLdVXdNGEycC8iUh0Vdk=	7bJBytvHV5+ydgof8ciRE/+ZnjN7Toy6pjBDkq464B4=
InM9VhkXM3faImPnxaBP0A==	UJLeC18Wi0mC1Qum25YpL	filename||sha1	FdQORw/ILnQhCyBmIKZf3/prypL2auLE913CAZrpT41LuqLBxr2Uzm9Ocssme1Y=	pYPFN5F9qWwmHwIxDRS25LxFdp7VgrYvNqudO+2uS0g=
gjgxpVpK4aHj2FX+IfUCXw==	fm8B/PxTQdG1nCMS9V3RRv	filename||sha1	wXYDheg58b25gPwBCcSCW1BK7f9ZHH4A3XjUBWJ8LtfqQBgBEuu3JYqiMuUQw2s=	p9jQwt2fQHPtjNzfya+ILCXeYNbvhJ6ojtho0emKmW4=
1/Zsph4vj420/mnPG62bhQ==	6+Fg3SpmRfI2HR5W/7AYYw	filename||sha1	D7VT7M9WZQnd+RfskPINIOJhwe2CC67/G/JxFKC84fjAyvF2qhsb+DELqZqAIqY=	aBHPP7naMpzudWreYCN7uQrPte+rgu0GeyUdZc7O3Jk=



Arguments or Redis

match rule

rules

for each IOC

Get value to test

value
attributes types = t

Get rules of type t
(if type no loaded yet)

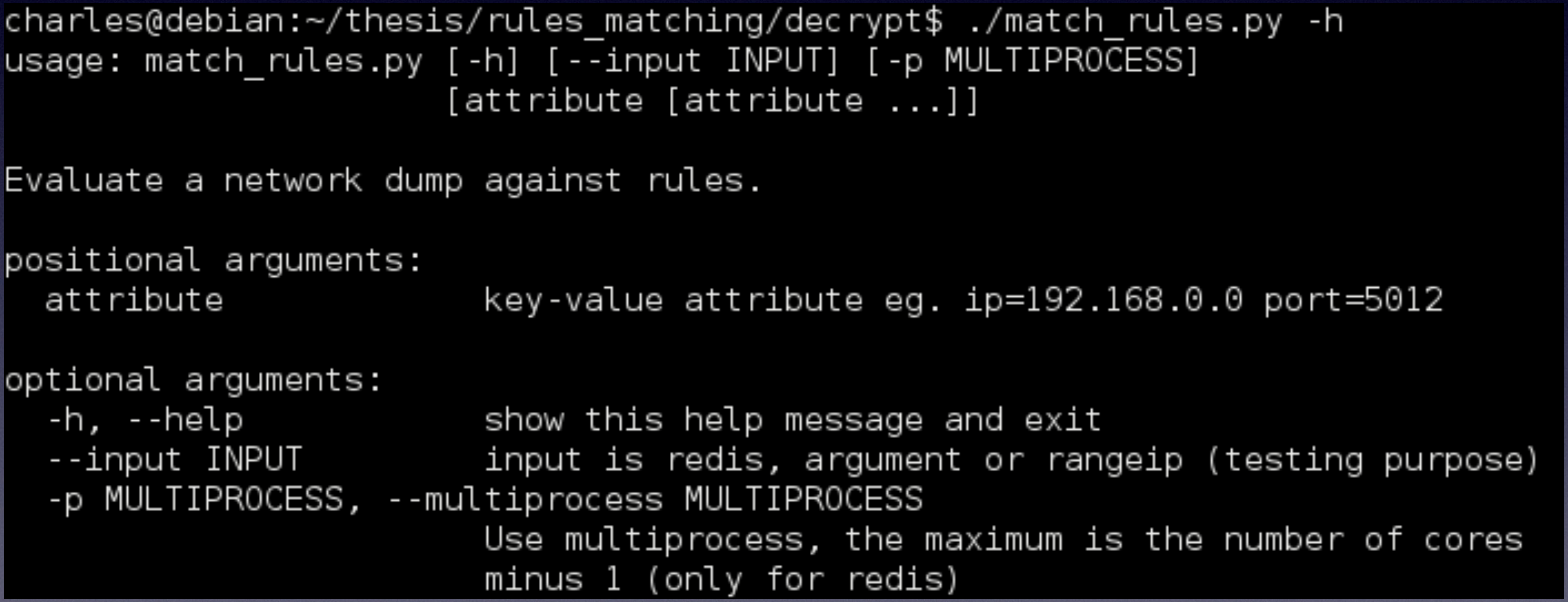
for each rule

Generate key k from attr and token

decrypt rule with k and salt

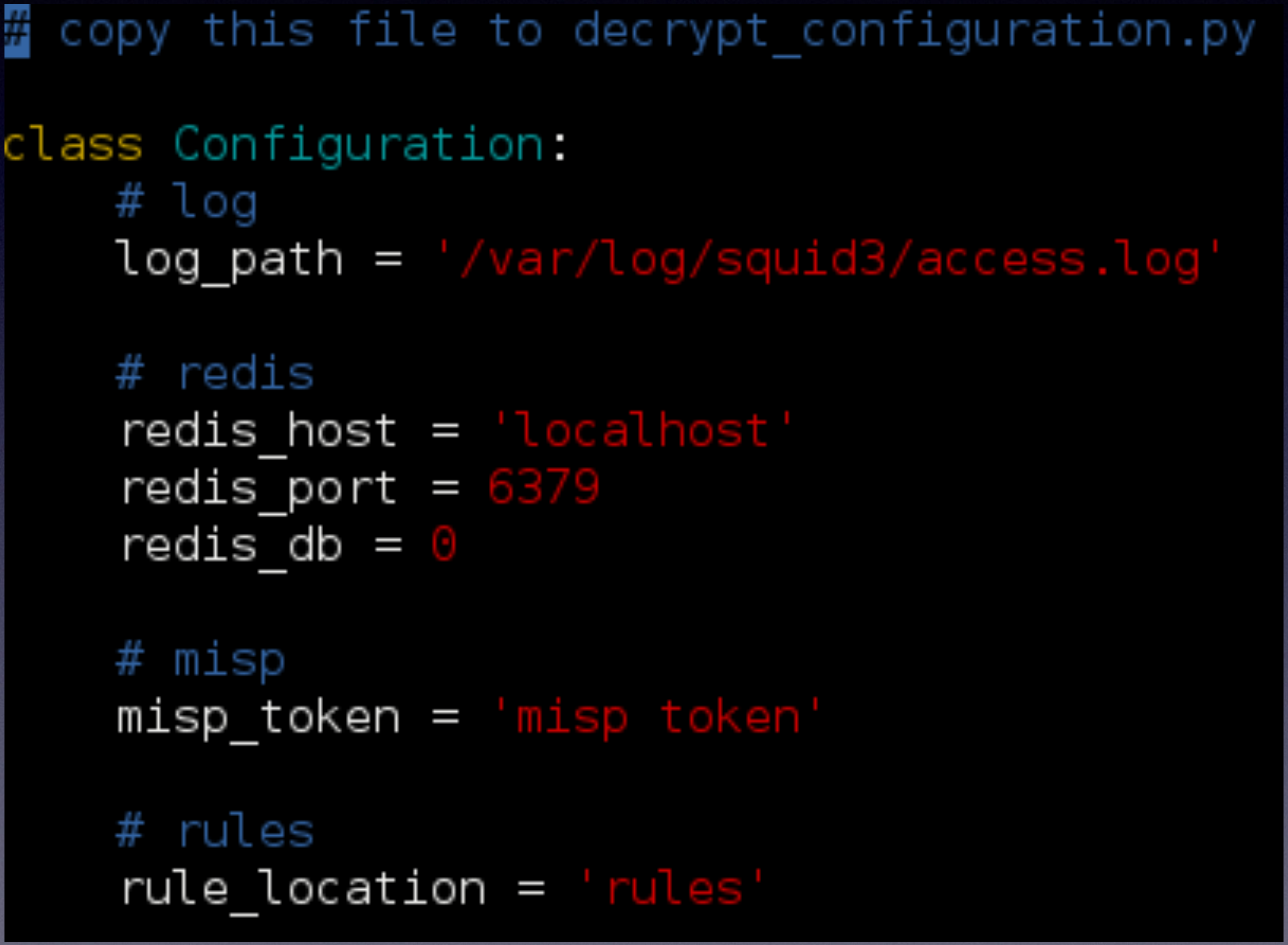
Check for a match

matchRules



```
charles@debian:~/thesis/rules_matching/decrypt$ ./match_rules.py -h
usage: match_rules.py [-h] [--input INPUT] [-p MULTIPROCESS]
                      [attribute [attribute ...]]

Evaluate a network dump against rules.

positional arguments:
  attribute             key-value attribute eg. ip=192.168.0.0 port=5012

optional arguments:
  -h, --help            show this help message and exit
  --input INPUT         input is redis, argument or rangeip (testing purpose)
  -p MULTIPROCESS, --multiprocess MULTIPROCESS
                        Use multiprocess, the maximum is the number of cores
                        minus 1 (only for redis)
```

Configuration

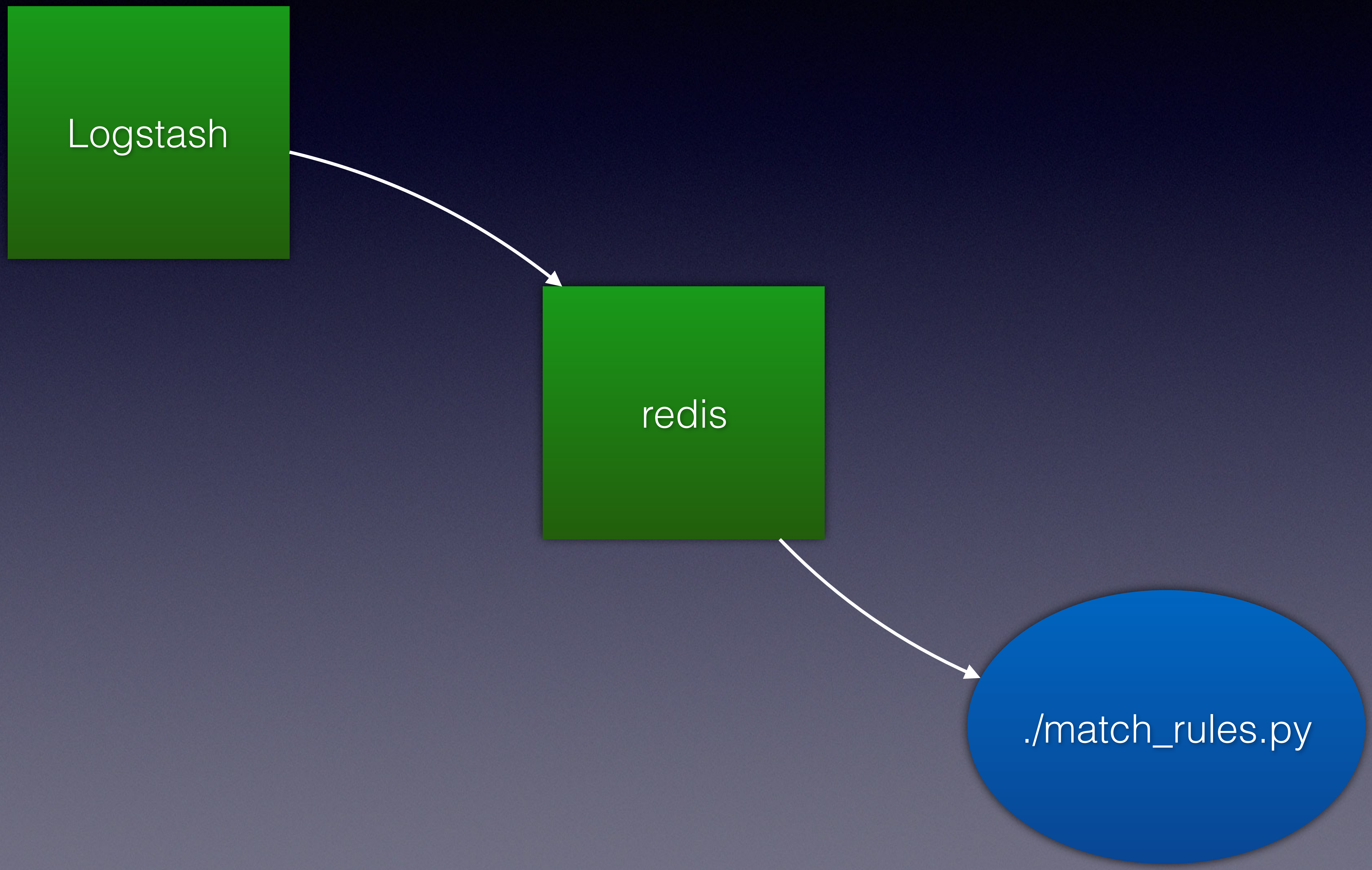


```
# copy this file to decrypt_configuration.py

class Configuration:
    # log
    log_path = '/var/log/squid3/access.log'

    # redis
    redis_host = 'localhost'
    redis_port = 6379
    redis_db = 0

    # misp
    misp_token = 'misp token'

    # rules
    rule_location = 'rules'
```

Url normalization

url_normalize (python library)
=> ..org

Pipe to read logs

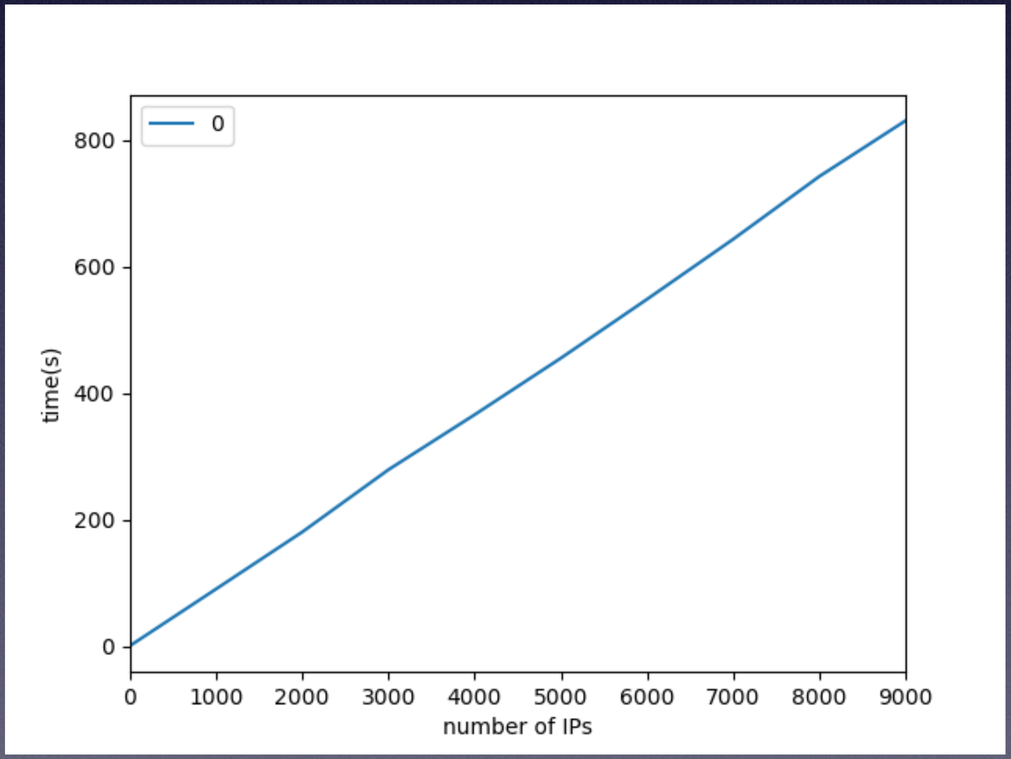


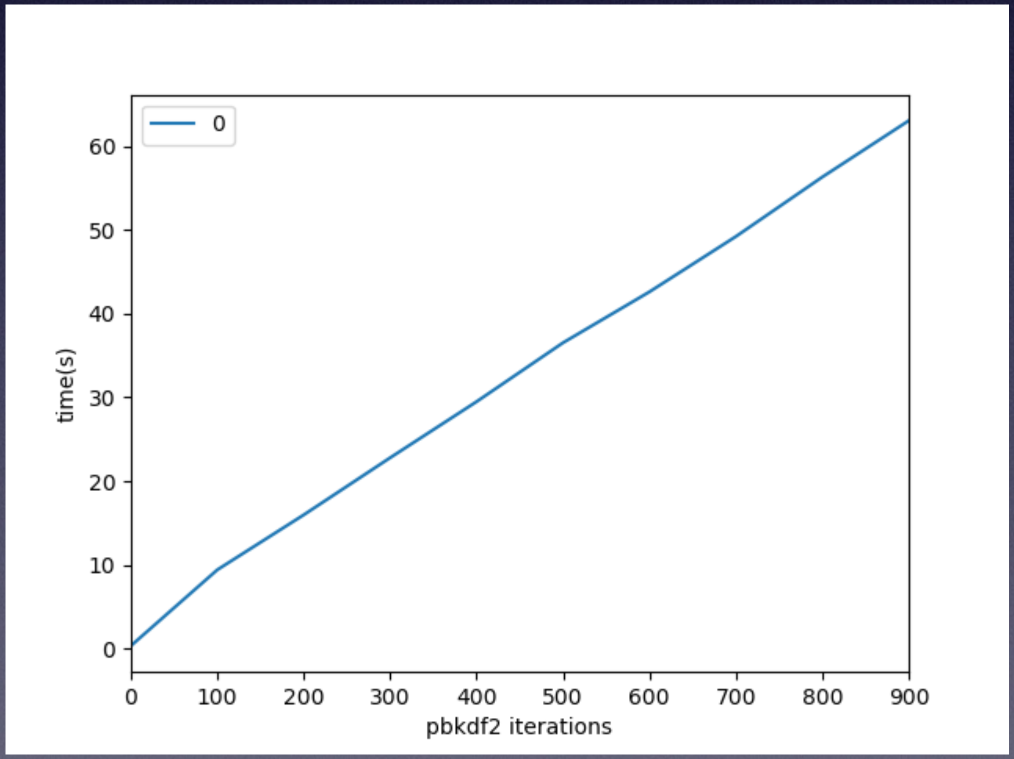
MultiProcessing

is enabled in the last version (if more than one element to check)

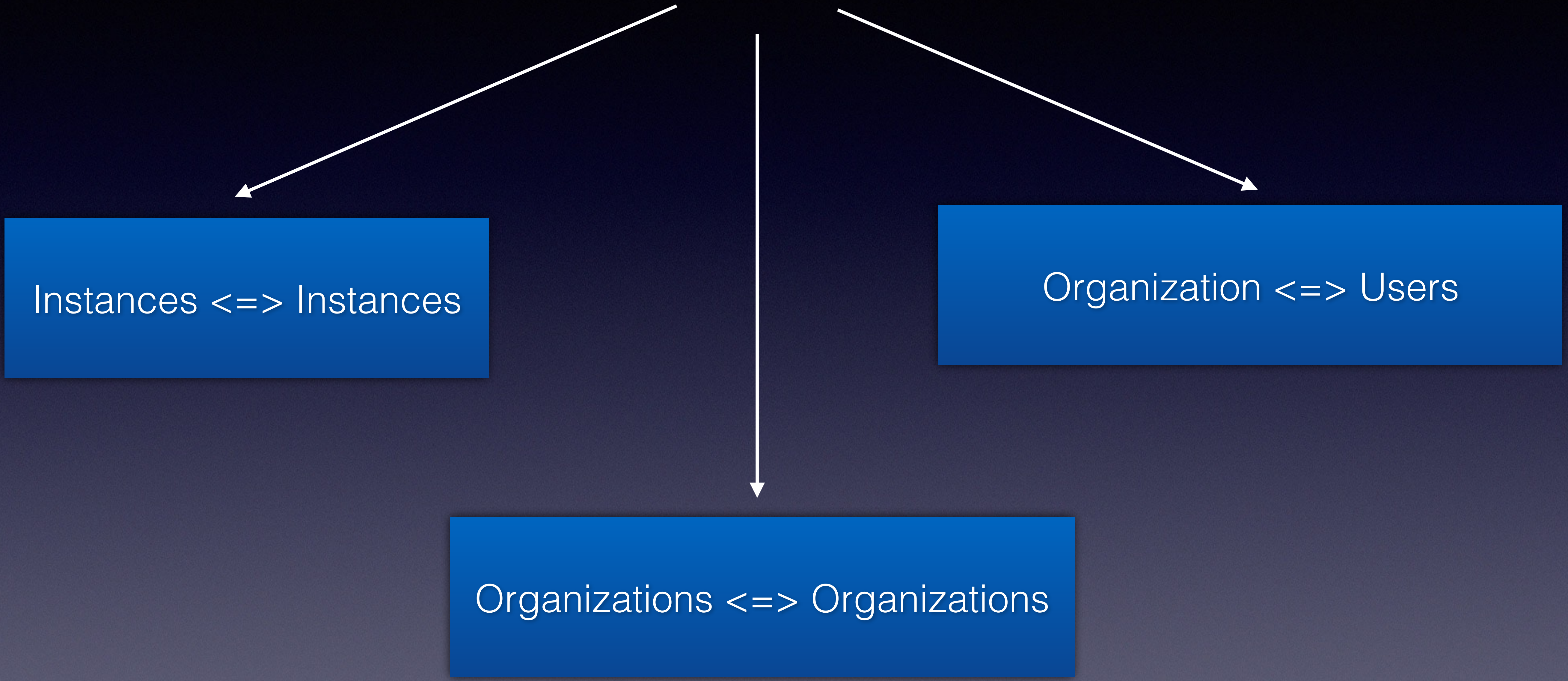
IP brute force

Try all ip in a small range 192.168.0.0/24





Sightings



Instances <=> Instances

Organization <=> Users

Organizations <=> Organizations

Sightings

Users <=> Organization

Trust ?

If the sightings is general ?

Sightings

Organizations <=> Organizations

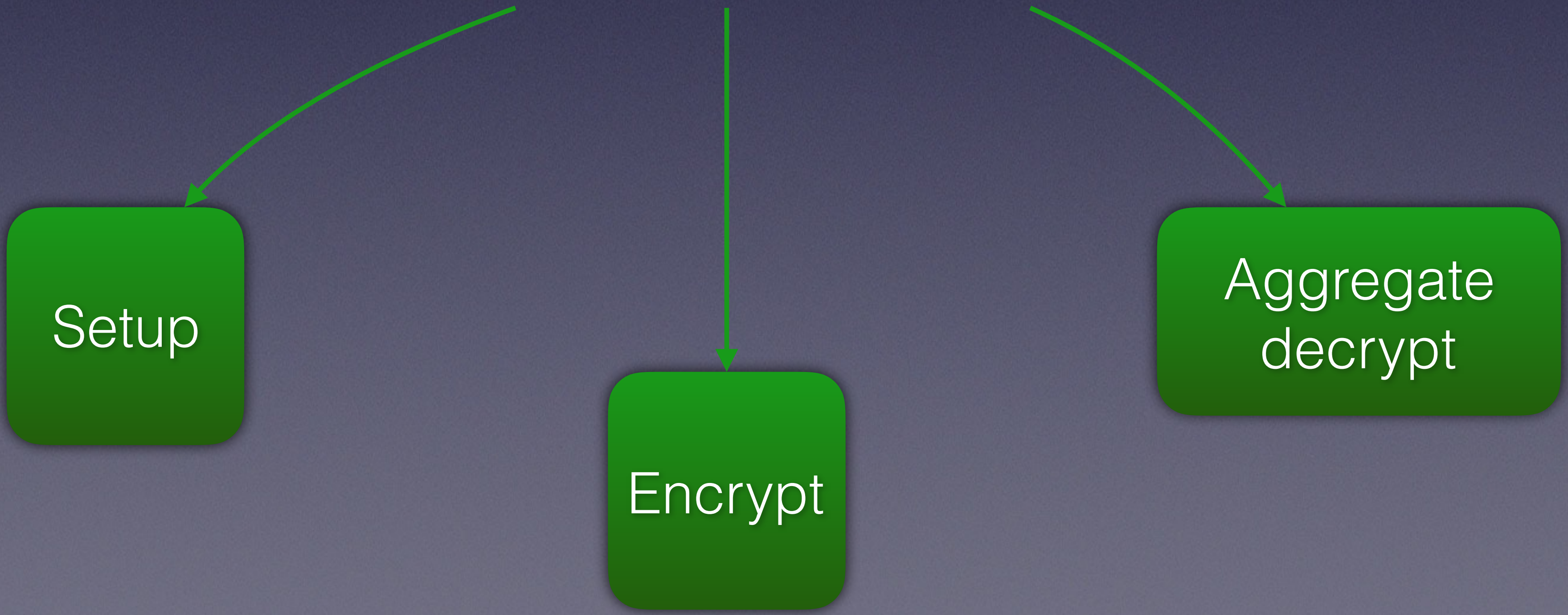
Directly in database ?

Sightings

Instances <=> Instances

=> can be done asynchronously?

=> Can implement the paper



Setup

Encrypt

Aggregate decrypt