

CHES 2017

Capture the Flag

Challenge

The WhibOx Contest

An ECRYPT White Box Cryptography Competition



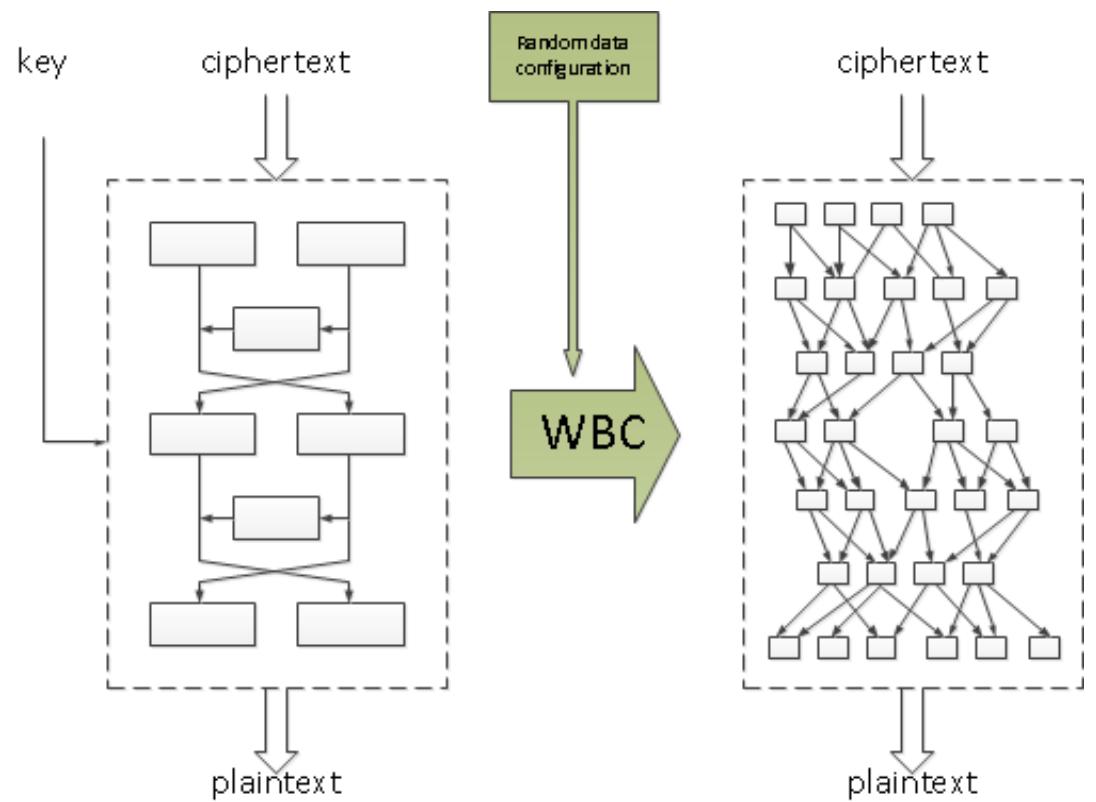
Organisation

- Organised by the ECRYPT-CSA project
- Submission server developed by CryptoExperts
- Submission server hosted by TU Eindhoven



White-box crypto

- Obfuscation for crypto implementations
- Should at least be secure against key extraction
- Every published scheme broken
- Big trend in the industry (mobile payment, DRM, ...)
- Deployed implementations based on secret technologies



(picture source: <http://www.whiteboxcrypto.com/>)

White-box contest

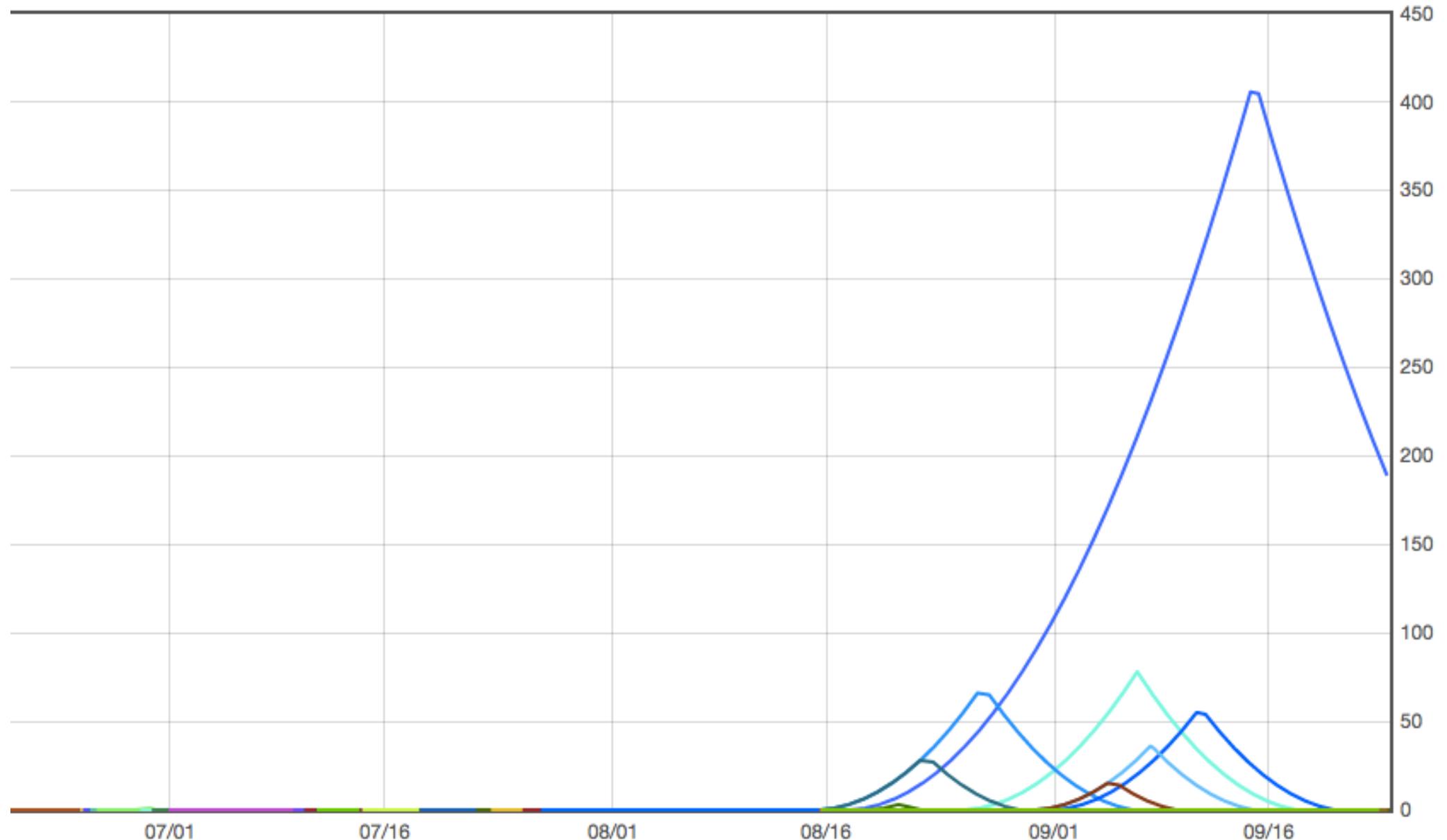
- Following an open discussion at the **WhibOx** workshop (co-located with CHES & CRYPTO 2016)
- Goal: confront designers and attackers of **practical white-box crypto**
- Designers can submit WB AES implementations st:
 - C **source code** at most **50MB**
 - **Executable** at most **20MB**
 - Use at most **20MB of RAM**
 - Run in at most **1sc**
- Attackers can try to recover the keys of submitted implem.

Contest rules

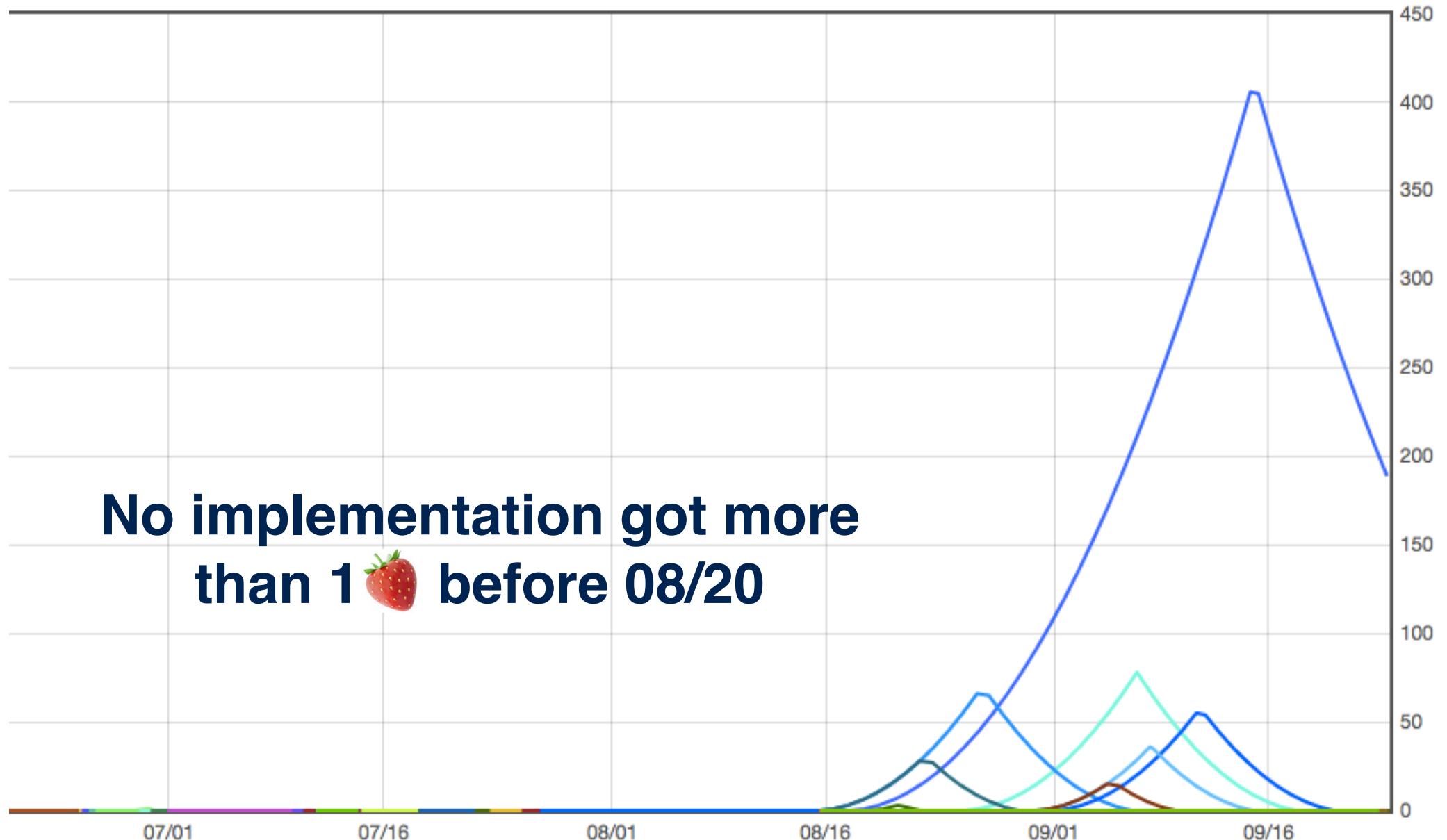


- A WB implem. gets points as long as it stays unbroken
- n new on day n (quadratic growth)
- When a WB implem. with q is broken:
 - the attacker gets q points (with max rule)
 - the score of the implem. starts to decrease symmetrically down to **0**
 - the designer of the implem. gets q points

Strawberry scores over time



Strawberry scores over time

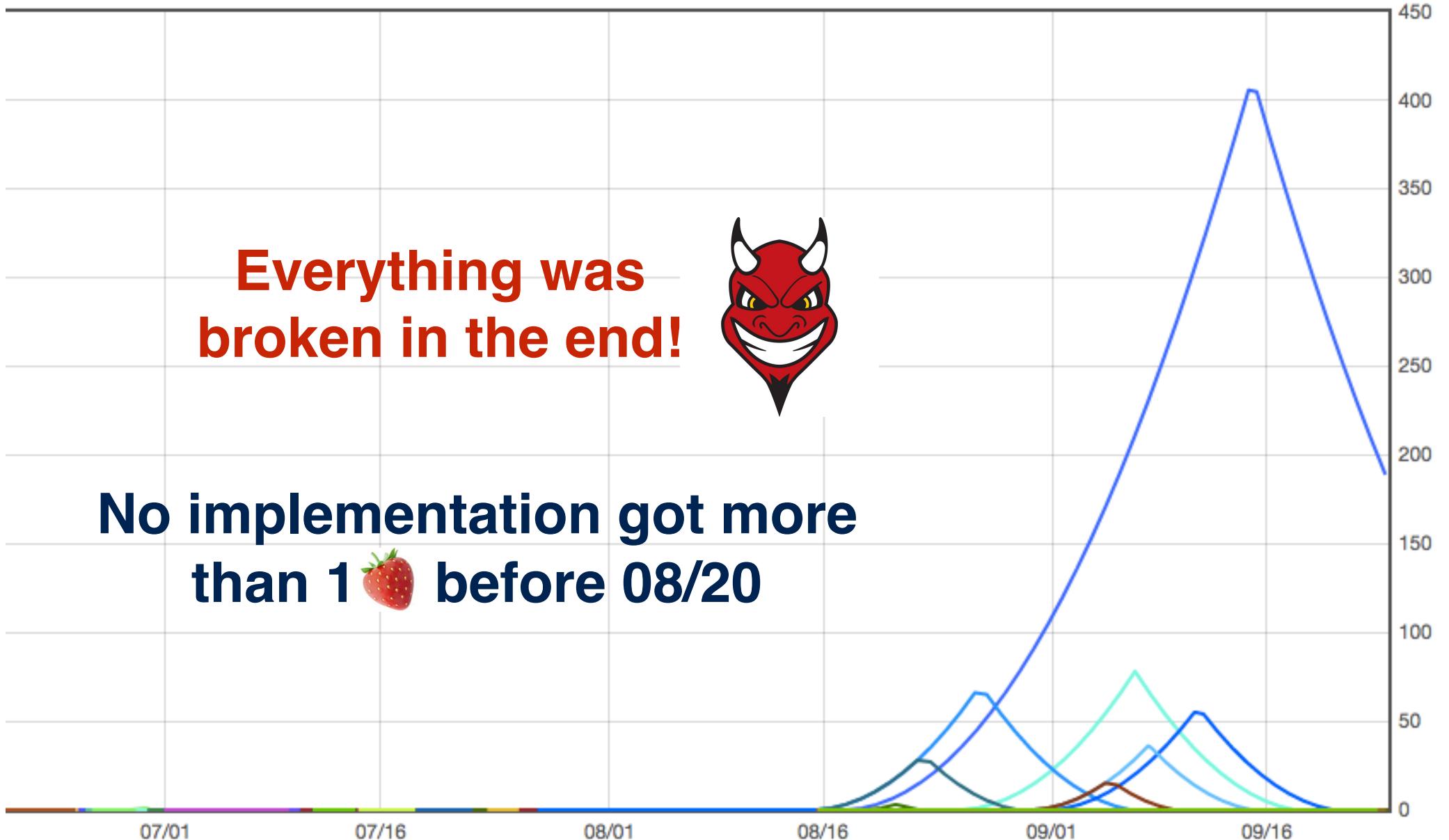


Strawberry scores over time

**Everything was
broken in the end!**



**No implementation got more
than 1 🍓 before 08/20**

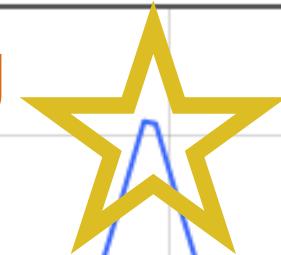


Strawberry scores over time

Everything was
broken in the end!



Outstanding
winner



No implementation got more
than 1 🍓 before 08/20

07/01

07/16

08/01

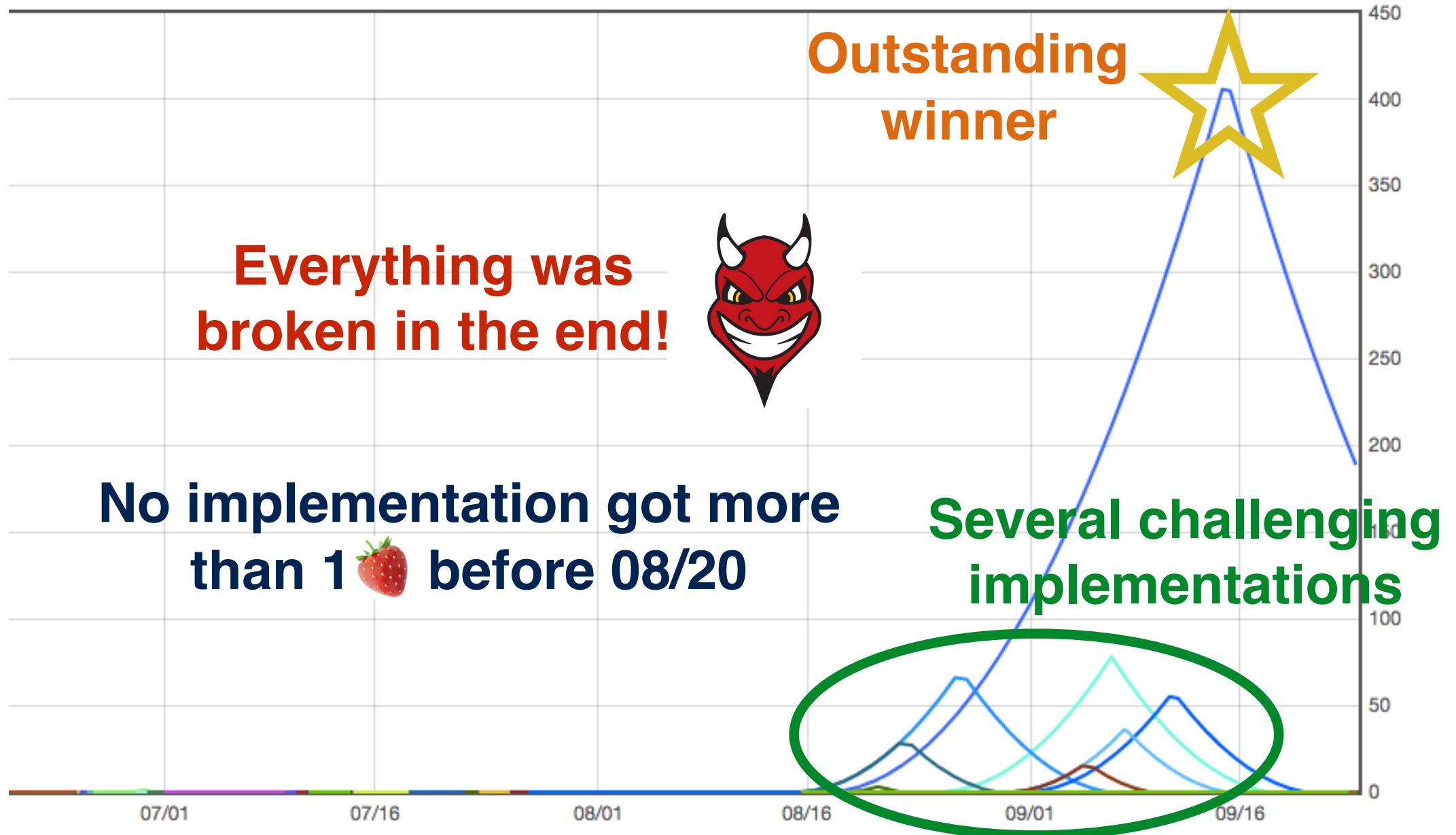
08/16

09/01

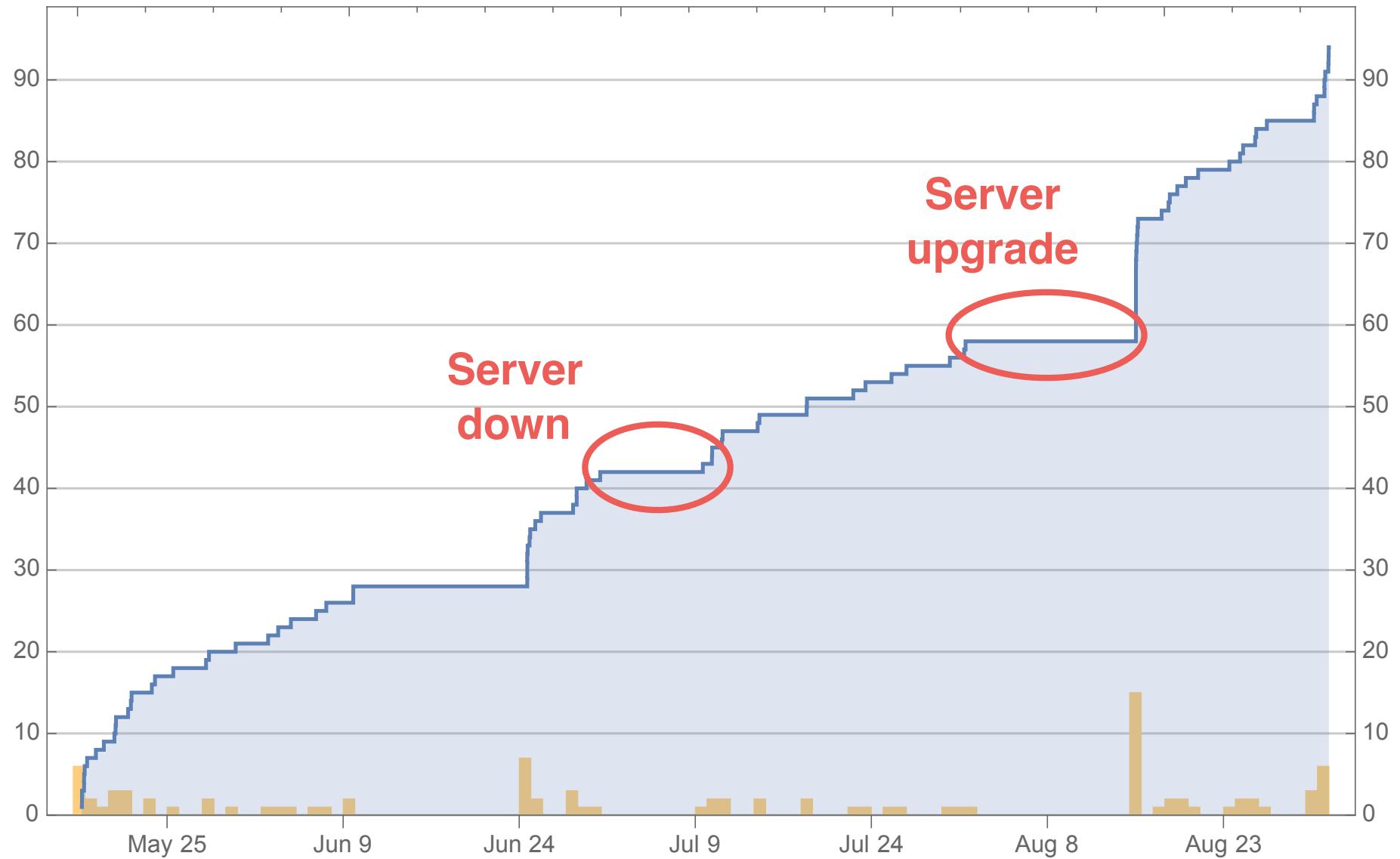
09/16



Strawberry scores over time



Submissions over time

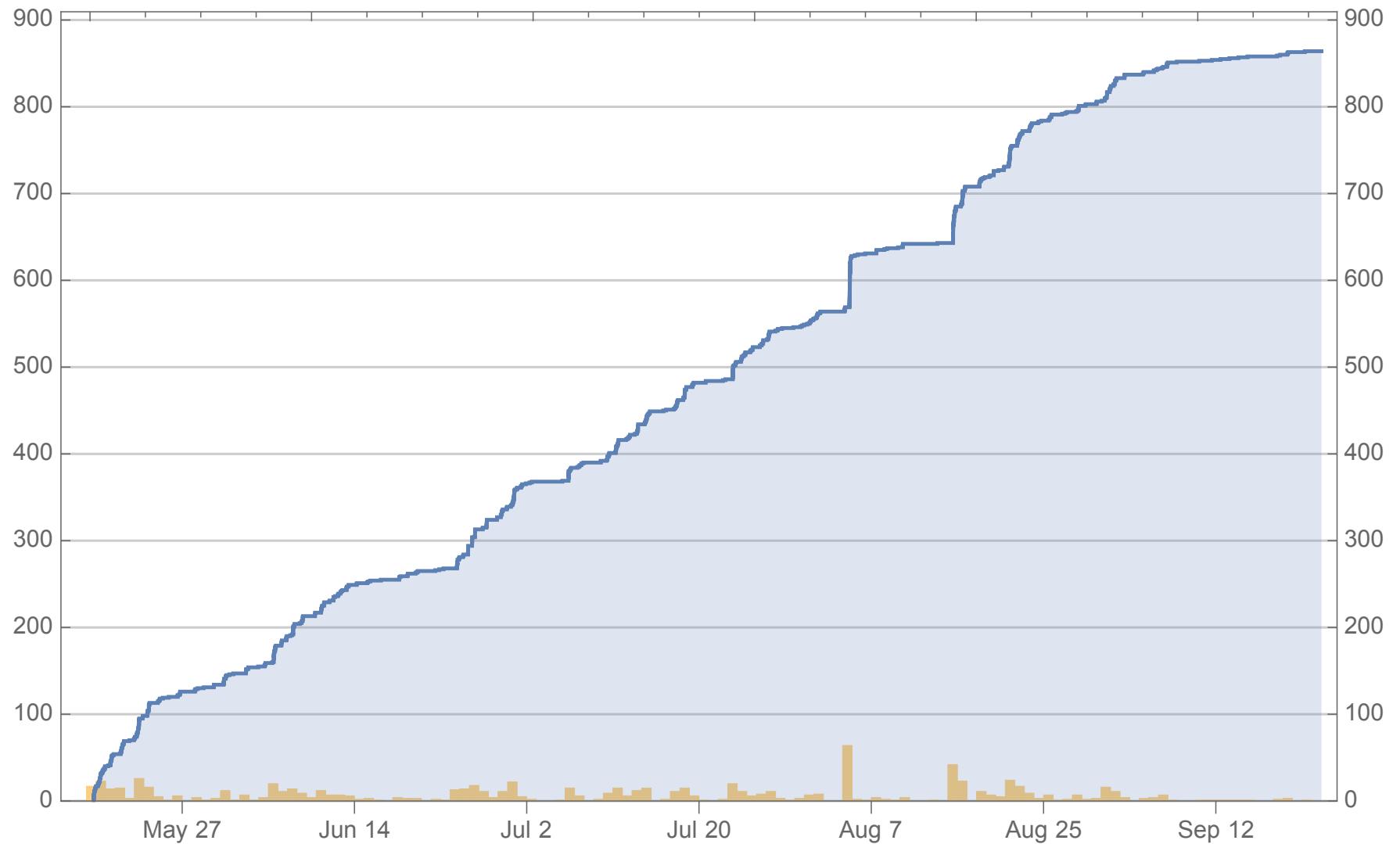


Submitters

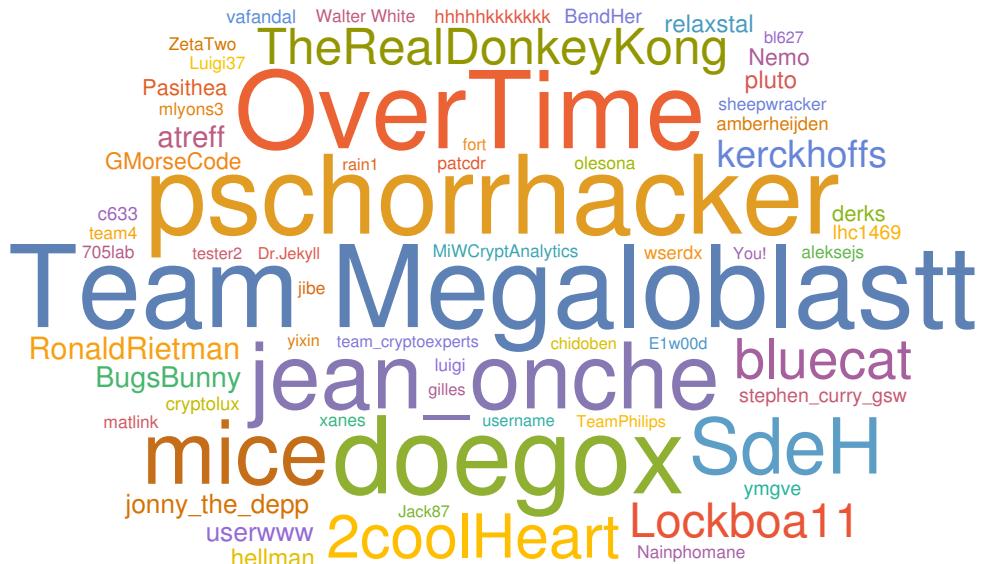
tester Guntram team4 olesona
fumiste Stradivarius grothendieck
JustTesting bluecat
T33 BugsBunny ssmiler
alec dequeue
uglyalgaes negainoido
aleksejs kluxc3qa1 lulu
cryptolux Milky
yixin yx Alt
chaes
GMorseCode
yixin therrealvalensteam Luigi37
Mr.Hyde sebastien-riou Feeker
coder1987 geronimo Qe1d28d67
double BendHer Nemo test1234

20	chaes	1	uglyalgaes
7	kluxc3qa1	1	tester
7	GMorseCode	1	test1234
5	negainoido	1	T33
4	bluecat	1	Stradivarius
4	alec	1	Qe1d28d67
3	therrealvalensteam	1	olesona
3	ssmiler	1	Mr.Hyde
3	BugsBunny	1	Milky
2	yixin	1	Luigi37
2	team4	1	JustTesting
2	sebastien-riou	1	Guntram
2	Nemo	1	grothendieck
2	lulu	1	fumiste
2	geronimo	1	double
2	Feeker	1	dequeue
2	coder1987	1	cryptolux
2	aleksejs	1	BendHer
1	yx	1	Alt

Number of breaks over time

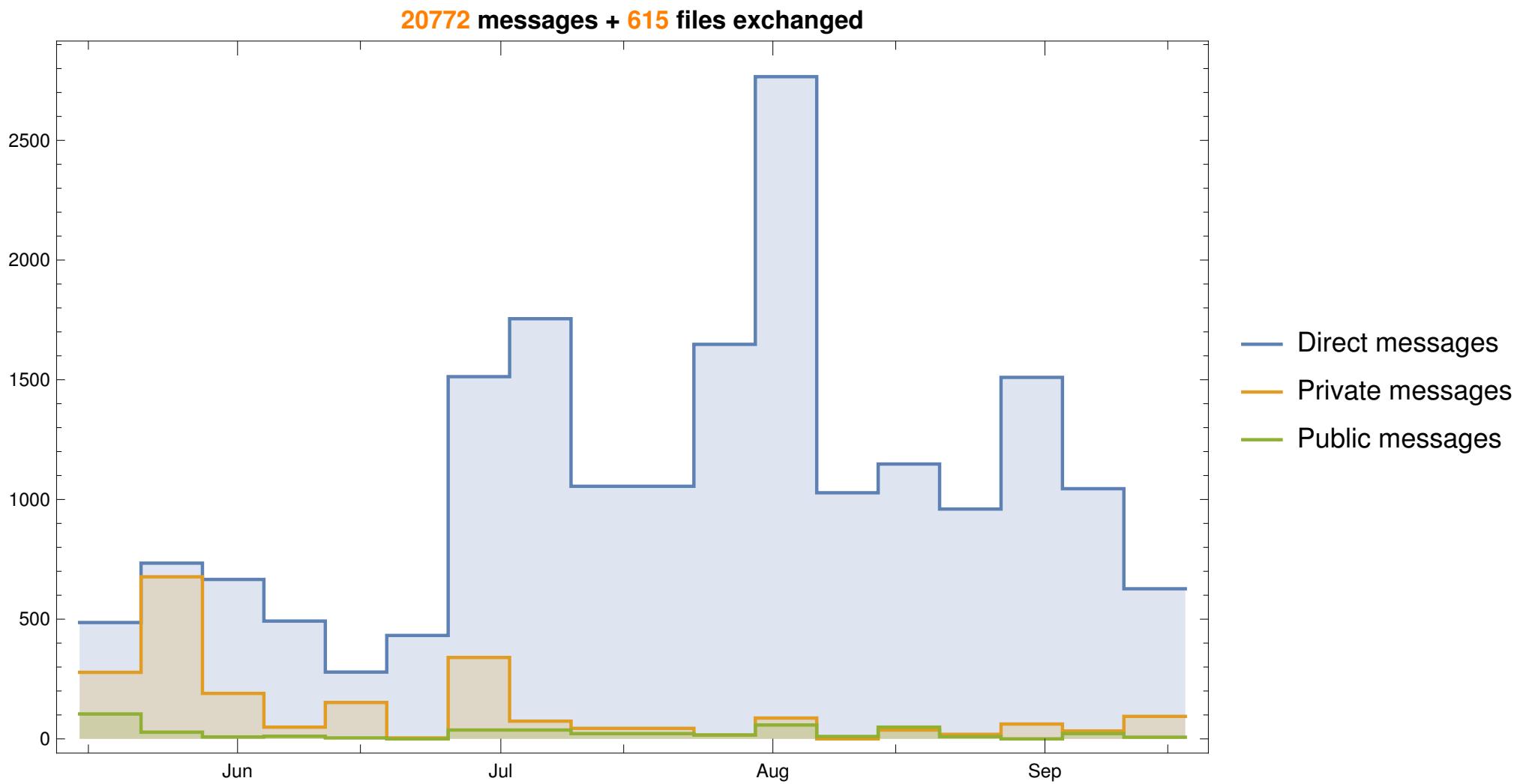


Breakers



89	Team Megaloblastt	3	c633
85	pschorrhacker	3	amberheijden
83	OverTime	2	ZetaTwo
83	doegox	2	xanes
69	jean_onche	2	Walter White
66	mice	2	team4
56	SdeH	2	sheepwracker
41	2coolHeart	2	patcdr
36	bluecat	2	olesona
31	TheRealDonkeyKong	2	Nainphomane
29	Lockboa11	2	mlyons3
20	kerckhoff	2	MiWCryptAnalytics
14	RonaldRietman	2	matlink
13	atreff	2	Luigi37
12	BugsBunny	2	luigi
10	userwww	2	jibe
10	jonny_the_depp	2	aleksejs
8	relaxstal	2	705lab
8	pluto	1	You!
8	hellman	1	yixin
7	Nemo	1	username
7	GMorseCode	1	tester2
6	Pasithea	1	TeamPhilips
6	derks	1	team_cryptoexperts
4	ymgve	1	rain1
4	stephen_curry_gsw	1	Jack87
4	lhc1469	1	gilles
4	BendHer	1	fort
3	wserdx	1	E1w00d
3	vafandal	1	Dr.Jekyll
3	hhhhkkkkkkk	1	chidoben
3	cryptolux	1	bl627

Slack activity



Challenges were broken **9.33** times on average

#Breaks	Most broken
39	wizardly_shannon
37	angry_meitner
20	hopeful_liskov quirky_keller
18	elegant_sinoussi
16	stupefied_varahamihira
14	famous_stonebraker practical_cori
13	eloquent_indiana
12	festive_jennings modest_clarke zealous_ardinghelli determined_goldwasser nostalgic_noether vigilant_heyrovsy

#Breaks	Least broken
5	mystifying_galileo silly_feynman priceless_stallman relaxed_allen musing_lalande compassionate_albattani smart_ardinghelli angry_jones nervous_montalcini
3	sad_goldstine bright_morse
2	relaxed_brown hungry_clarke vibrant_goldberg
1	jolly_davinci competent_agnesi adoring_poitras

Challenges were broken **9.33** times on average

#Breaks	Most broken
39	wizardly_shannon
37	angry_meitner
20	hopeful_liskov quirky_keller
18	elegant_sinoussi
16	stupefied_varahamihira
14	famous_stonebraker practical_cori
13	eloquent_indiana
12	festive_jennings modest_clarke zealous_ardinghelli determined_goldwasser nostalgic_noether vigilant_heyrovsy

#Breaks	Least broken
5	mystifying_galileo silly_feynman priceless_stallman relaxed_allen musing_lalande compassionate_albattani smart_ardinghelli angry_jones nervous_montalcini
3	sad_goldstine bright_morse
2	relaxed_brown hungry_clarke vibrant_goldberg
1	jolly_davinci competent_agnesi adoring_poitras

Winner (28 days / 406) 

Third (11 days / 66) 

Second (12 days / 78) 

Strawberry scoreboard

Rank			Name	Strawberries	
#	id			Peak	User
#1	777		adoring_poitras	406 	cryptolux
#2	815		competent_agnesi	78 	grothendieck
#3	753		bright_morse	66 	sebastien-riou
#4	877		vibrant_goldberg	55 	chaes
#5	845		hungry_clarke	36 	team4

Strawberry scoreboard

Rank			Name	Strawberries	
#	id			Peak	User
#1	777	■	adoring_poitras	126 🍓	cryptolux
#2	815	■	compr	106 🍓	hendieck
#3	753	■	bright_m	106 🍓	sebastien-riou
#4	877	■	vibrant_goldberg	55 🍓	chaes
#5	845	■	hungry_clarke	36 🍓	team4

Winners:
Alex Biryukov
Aleksei Udovenko
(U. Luxembourg)

Strawberry scoreboard

Rank			Name	Strawberries	User
#	id		Name	Peak	User
#1	777	■	adoring_poitras	406 🍓	cryptolux
#2	815	■	competent_agnesi	351 🍓	grothendieck
#3	753	■	bright_moon	333 🍓	sebastien-riou
#4	877	■	vibrant_goldberg	33 🍓	chaes
#5	845	■	hungry_clarke	36 🍓	team4

Still
anonymous

Strawberry scoreboard

Rank			Name	Strawberries	User
#	id			Peak	
#1	777	■	adoring_poitras	406 🍓	cryptolux
#2	815	■	competent_agnesi	78 🍓	grothendieck
#3	753	■	bright_morse	66 🍓	sebastien-riou
#4	877	■	Brent Carmer, Tancrède Lepoint, Alex Malozemoff, Mariana Raykova (iO with degraded parameters)	4 🍓	chaes
#5	845	■	team4	0 🍓	team4

Banana scoreboard

Rank	User	Bananas
#1	team_cryptoexperts	406 🍌
#2	cryptolux	78 🍌
#3	You!	55 🍌
#4	Team Megaloblastt	44 🍌
#5	jean_onche	28 🍌

Banana scoreboard

Rank	User	Bananas
#1	team_cryptoexperts	406 🍌
#2	cryptolux	38 🍌
#3	You!	36 🍌
#4	Team Megalobid	44 🍌
#5	jean_onche	28 🍌

Winners:
**Louis Goubin, Pascal Paillier,
Matthieu Rivain, Junwei Wang
(CryptoExperts)**

Banana scoreboard

Rank	User	Bananas
#1	team_cryptoexperts	406 🍌
#2	cryptolux	
#3	You!	
#4	Louis Matthijs	
#5		



Louis
Matthijs

We didn't cheat!

- * We didn't host the server
- * Sources on GitHub
- * Junwei's presentation next
- * White-paper coming soon

Banana scoreboard

Rank	User	Bananas
#1	team_cryptoexperts	406 🍌
#2	cryptolux	78 🍌
#3	You!	55 🍌
#4	Team Megaloblastt	44 🍌
#5	jean_onche	28 🍌

Alex Biryukov
Aleksei Udovenko
(U. Luxembourg)

Banana scoreboard

Rank	User	Bananas
#1	team_cryptoexperts	406 🍌
#2	cryptolux	78 🍌
#3	You!	
#4	Team Megaloblastt	
#5	jean_onche	28 🍌

Still
anonymous

Reveal Secrets in Adoring Poitras

A victory of reverse engineering and cryptanalysis over challenge 777

Louis Goubin^{1,4} Pascal Paillier¹
Matthieu Rivain¹ Junwei Wang^{1,2,3}

¹CryptoExperts

²University of Luxembourg

³University of Paris 8

⁴University of Versailles-St-Quentin-en-Yvelines

CHES 2017, Rump Session, Taipei

Outline

- 0.** ■ Downloading and Compiling the Code
- 1.** ■ Cleaning the Code
- 2.** ■ De-Virtualization
- 3.** ■ From Bitwise Program to Boolean Circuits
- 4.** ■ Boolean Circuits Minimization
- 5.** ■ Data Dependency Analysis
- 6.** ■ Algebraic Analysis

Outline

- 0 ■ Downloading and Compiling the Code**
- 1 ■ Cleaning the Code**
- 2 ■ De-Virtualization**
- 3 ■ From Bitwise Program to Boolean Circuits**
- 4 ■ Boolean Circuits Minimization**
- 5 ■ Data Dependency Analysis**
- 6 ■ Algebraic Analysis**

Downloading and Compiling the code Code

- Browsers stuck at loading it...
- Editors are broken by it...
- Some compilers (e.g., llvm) keep compiling and reporting warnings...

 11:55 PM
777 broke my editor 😞

 8:14 AM
So was my compiler.

Outline

0. ■ Downloading and Compiling the Code

1. ■ Cleaning the Code

2. ■ De-Virtualization

3. ■ From Bitwise Program to Boolean Circuits

4. ■ Boolean Circuits Minimization

5. ■ Data Dependency Analysis

6. ■ Algebraic Analysis

Untidy Code

More than 1k functions

```
void x5nEq(uint UPMNsVlp, uint KtFY, uint vzJZq) {if(nIlajqq()==IFWBUN(UMNsVlp, KtFY)) Ewwon(vzJZq);}
void qfGrcRsk() {kIKfgI=bPcrtK());}
void mCazni(uint EwREsc, uint CLoJ) {if(Brza)==v~=
oid SNGxNy(uint lntfP, uint iLRoIt, uint KTf)
void rNUIPyD(uint hFqeoI, uint jvxpt) {xkpR
void vDlB(uint QRFdI, uint CoCI, uint aLPxx)
void UFjtwb(uint HCCoI, uint ISRFdIp, uint urrxX) {xkpR{<173937>}&ooGoRv[{kIKfgI+aLPxnn
void uJTr(uint RouDUC, uint TSCaTL) {return ooGoRv[763216u1]};qscwtK(RouDUC+(kIKfgI<<17),TSCaTL);}
void WnCwUs(uint AkUlfh, uint xtva) {xkpR{AkUlfh}=xkUluIdu(xtva);}
void ckn(uint ZxWn, uint dywdwc) {return ooGoRv[{ZxWn+dywdwg}|196983];}
uint mgxgNy(uint cdq, uint aLEjS) {return TzXsSE(cdq+(kIKfgI>>18),aLEjS);}
void gtdkbbX(uint OIModr, uint TuqiqX) {xkpR{OIModr}=&uNqT{TuqiqX};}
void TbsWl(uint WTApxRg, uint Wnx) {VdpRwo=ooGoRv[MYArFxG]&Wnx;}
void puBLD(uint XBzm, uint aZGHRL) {xpkJt(XBzm, PuixLaZGRHL);}
uint Mpfn(uint ghpcTsG, uint UeGvxVS) {return nPLEFT(ghpcTsG+(kIKfgI<<18),UeGvxVS);}
void zsBuI(uint NCJW, uint OMOPBxq, uint AyolFz, uint BwxtvX) {uint dyfs=(ooGoRv[{kIKfgI+AyoFz}&14262143]>&BwxtvX)&1;ooGoRv[{kIKfgI
void CNul(uint UtDajEs, uint VktOYZ) {ooGoRv[{kIKfgI+UtDajEs}|113092]>&ooGoRv[{kIKfgI+VktOYZ}&8473|j])
void ZAPuR(uint manfh, uint Gziw) {ooGoRv[{kIKfgI+manfh}|128149]>&ooGoRv[{((ooGoRv[{kIKfgI+Gziw}|142194])|16981)<<1}>437+(((
void eoYLqcs(uint GaULgs, uint xVRz, uint bTuaFhw) {if(jQdGL)=Mpfn(GaULgs, xVRz)} Ioez(bTuaFhw);}
void ghalu(uint WtHze, uint HmAOVm, uint xAXd0) {ooGoRv[{kIKfgI+EfrHz}&262143]=<>ooGoRv[{kIKfgI+HmAOVm}&262143]>>57}&(ooGoRv[(
void bqlseRp(uint Ktkq, uint ggaA5, uint aPrcDv) {ooGoRv[{kIKfgI+ktkq}|167828]=&ooGoRv[(&HmAOVm+qAS)&196972]}&ooGoRv[{kIKfgI+al
void veUmt(goRv, uint spuKo) {return ooGoRv[onyGwJpVxO|262143];}
void XCHfHe(uint IXxBG, uint BlPTTU) {VdpRwo-&ooGoRv[IXxBG|BlPTTU];
void HKYPrQ(uint ZnDy, uint udNfCp, uint JtGK) {ooGoRv[{kIKfgI+JtGK}|262143];
void dnEmK(uint WAdv, uint ZcvD) {return ooGoRv[{WAdv+ZcvD}|&262143];}
void Q1Ix0(uint KledyCw, uint LrjmjY) {luDBCn(KledyCw, xkUluIdu(LrjmjY-kIKfgI));}
void koPlBY(uint wamXceM) {kIKfgI=wamXceM;}
void BndSNHO(uint efusN, uint qSia) {ooGoRv[{kIKfgI+efusN}|211401]=&ooGoRv[{kIKfgI+qSia}&129545]<<25;};
void zuWrmHs(uint MnjJkN, uint LZKwN) {ooGoRv[{kIKfgI+MnjJkN}|230698]=&ooGoRv[{kIKfgI+LzKwN}&626145]>>9;};
void TUnc(uint DqPwv, uint chNeV, uint xtFa) {ooGoRv[{kIKfgI+DqPwv}&262143]=&ooGoRv[{kIKfgI+chNeV}&262143]<<23}>>ooGoRv[(
void Euc(tl, uint vise, uint vise) {lDgabWw(hrPwv, InGSm{vEyy+kIKfgI});}
void dZjbPw(uint Ynf, uint JKTw) {ooGoRv[{kIKfgI-Ynf}&262143]=&ooGoRv[{kIKfgI+jKTw}&262143];
void UtRnRgD(uint Rgn, uint Pgn, uint Hgn) {((ooGoRv[{kIKfgI-Ygn}&1493]&EmoKg_0gn)&Ioez(bUshRnRgD);}
void qDyIT0(uint mzIxRVL, uint LxTNtj) {ooGoRv[{kIKfgI+mzIxVL}|2263567]=&ooGoRv[{kIKfgI+LxTNtj}|200567]<<8;};
void seclu(uint mztrvI, uint FzTgat) {xkpR{mztrvI}=GCHMT{FzTgat};}
void KPKR() {KIKfgI=lrbvFt();}
void WgtDy(uint vgOlh, uint II1IKxA) {xkpR{vgOlh}=ruPEW(II1IKxA);}
void FPHPLXL(uint StEqD, uint wntBEP, uint ZlQ) {ooGoRv[{kIKfgI+StEqD}&1943];
uint SCxxCkL(uint TPoff, uint McOb) {return ooGoRv[{TPoff+McOb}&1943];
void mQheVs(uint wwyJcD, uint yJcD, uint GAGxCL) {if(jQdGL)=HvD
uint feTzo) {return fQyyDK({kIKfgI<<18}+1592, (VdpRwo>>6)<<(VdpRwo
void MOAi(uint Yehp, uint yuiqcAh) {VdpRwo=ooGoRv[YehpB]|yuiqcAh; r
void vGqvMu(uint fChT) {kIKfgI = fChT;}
void wnaEglai(uint a0jFaul, uint DdwxiQu) {ooGoRv[{kIKfgI+a0jFaul}&262143]&DdwxiQu;}
void pcesgI(fTcT)()
void oldxtwXfZl(uint ltnK, uint zhuxy) {ooGoRv[{kIKfgI+loTKN}&262143]=&ooGoRv[{kIKfgI+zMuxy}&262143];}
void gruChs() {return fcOpRk(kIKfgI>>10)|077, (VdpRwo>>1)<<1148372;}
```

Readability Processing

- Duplicate / redundancy / unused codes elimination
- Functions / variables renaming
- Constants rewriting
- Code combination

Readability Processing

- Duplicate / redundancy / unused codes elimination
- Functions / variables renaming
- Constants rewriting
- Code combination

Only 20 functions are remaining

```
void copy(uint KLedyCW, uint lRjmjY) {int_arr[(a+KLedyCW) & 0x3ffff] = int_arr[lRjmjY & 0x3ffff];}
void encode(uint owhj0, uint nlBqXn) {assign(owhj0, in_ptr[nlBqXn]);}
void decode(uint hFqeIO, uint jvXpt) {out_ptr[hFqeIO]=int_arr[(a+jvXpt)&0x3ffff];}
void rshift_xor(uint HCOL, uint ISRFdIp, uint uFYFMX) { int_arr[HCOL&0x3ffff]^=1&(int_arr[(a+ISRFdIp)&0x3ffff]>>ufYFMX); /*pri
void lshift_xor(uint NCjBw, uint OMQBXqa, uint AyoLFz) { uint dyf5=(int_arr[(a+AyoLFz)&0x3ffff])&1;int_arr[(a+NcJBw)&0x3ffff]^=dyf5;
void expand_bit(uint SteQld, uint ZubEP, uint ZiQz) {int_arr[(a+SteQld)&0x3ffff]=~((int_arr[(a+ZubEP)&0x3ffff]>>ZiQz)&1);}

uint lookup1(uint AKBKig) {return int_arr[(a+AKBKig)&0x3ffff];}
uint lookup2(uint WAdV, uint ZcVdJ) {return int_arr[(WAdV-ZcVdJ)&0x3ffff];}

void assign(uint UbEJi, uint UmwjUh0) {int_arr[(a+UbEJi)&0x3ffff]=UmwjUh0;}
void assign_a(uint WE0kx) {a = WE0kx;}
void assign_b(uint fnmqxL) {b=int_arr[fnmqxL]&0x0ffff;}
void update_a() {a=lookup2(1592,mix(b)); printf("%lu\n",a);}
void update_b() {b=0x7fff&lookup2(522,mix(b));}

void mystery(uint wJxeA, uint QBGXUN) {uint t = (~int_arr[(a+QBGXUN)&0x3ffff])&0x7fff; assign(wJxeA,lookup2(2979,mix(t)));}

// bitwise operation
void xor(uint oEHmwk, uint KCZu, uint MtCA) {int_arr[(a+oEHmwk)&0x3ffff]=int_arr[(a+KCZu)&0x3ffff]^int_arr[(a+MtCA)&0x3ffff];}
void and(uint bmmFp, uint UNFg, uint PqCtYZ) {uint t = int_arr[(a+UNFg)&0x3ffff]&int_arr[(a+PqCtYZ)&0x3ffff];}
void or(uint eTGI, uint udoxFs, uint mezPNN) {int_arr[(a+eTGI)&0x3ffff]=int_arr[(a+udoxFs)&0x3ffff]|int_arr[(a+mezPNN)&0x3ffff];}
void not(uint YfnT, uint JKTW) {int_arr[(a+YfnT)&0x3ffff]=~int_arr[(a+JKTW)&0x3ffff];}

// jump
void goto_f(uint LKhOC) {pc = bop + LKhOC;}
void jump_if(uint DbvJO, uint FleFNIf, uint LeHf) { if(lookup2(2979,mix(b))==lookup2(DbvJO, FleFNIf) || count >= 64) {printf("%d"

```

Outline

- 0.** ■ Downloading and Compiling the Code
- 1.** ■ Cleaning the Code
- 2.** ■ De-Virtualization
- 3.** ■ From Bitwise Program to Boolean Circuits
- 4.** ■ Boolean Circuits Minimization
- 5.** ■ Data Dependency Analysis
- 6.** ■ Algebraic Analysis

Universal Turing Machine

```
2262     uint kIKfgI, VDpRwo;
2263     uchar *JqcadL;
2264     uchar *xkpRp, *Puix;
2265     uchar *sutlnu;
2266     uchar *unYAQ;
2267     const char pDeoW[] = "öSTXUS}BS&yÖ:eK±Ízüli+LžÄpē<Üþç+Àå]'*0DC3¾ŽBElttS0ávvÝþK,,ÓStä±K_Gâax    FTBzif
2268     uint ooGoRv[sizeof(pDeoW)/sizeof(uint)];
2269     char JGNnvI[] = "S0\0\n\0\0\0\0\0\0STXSOHxSUBSOH\0\0\0\0\0\0\0\0\0\0\0\0STXSOHETX\0P\0\0\0\0\0\0\0\0\0\0\0\0\0\0";
2270     uchar *klspCVy = (uchar*)JGNnvI;
2271     void _ALLI(){
2272     unYAQ = klspCVy;
2273     sutlnu = klspCVy + sizeof(JGNnvI)/sizeof(uchar);
2274     JqcadL = klspCVy;
2275     while (JqcadL < sutlnu) {
2276         uchar eMmr = *JqcadL++;
2277         if (eMmr == 0) {
2278             void (*QiEb)();
2279             QiEb = (void*)funcptrs[*JqcadL++];
2280             uint *AnezsV = (uint*)JqcadL;
2281             JqcadL += eMmr*8;
2282             QiEb();
2283         }
2284         else if (eMmr == 1) {
2285             void (*QiEb)(uint);
2286             QiEb = (void*)funcptrs[*JqcadL++];
2287             uint *AnezsV = (uint*)JqcadL;
2288             JqcadL += eMmr*8;
2289             QiEb(AnezsV[0]);
2290         }
2291         ...
2292     }
2293     void AES_128_encrypt(uchar *OLjd, uchar *xzptiF) {
2294     for(kIKfgI = 0; kIKfgI < sizeof(ooGoRv)/sizeof(uint); kIKfgI++) ooGoRv[kIKfgI] = gBXW[kIKfgI];
2295     xkpRp = OLjd;
2296     Puix = xzptiF;
2297     if (sizeof(klspCVy))
2298         _ALLI();
2299     else
2300     1;// c;
2301 }
```

→ UTM(RASP)

Universal Turing Machine (2)

```
2262 uint kIKfgI, VDpRwo;
2263 uchar *JqcadL;
2264 uchar *xkRp, *Puix;
2265 uchar *sutlnu;
2266 uchar *unYAQ;
2267 const char pDeoW[] = "äSTXUS}BS&yö:eK±ÜlÜli+LzåPé<~Üþç+Aå]" + 0DC3%ZREFtt$0ávYpk,,05täK_Gåx  FTBif
2268 uint ooGoRv[sizeof(pDeoW)/sizeof(uint)];
2269 char JGNnvI[] = "SOH\0\0\0\0\0\0\0\0STXSOH\xSUBSOH\0\0\0\0\0\0\0\0\0\0\0\0\0STXSOHETX\0P\0\0\0\0\0\0
2270 uchar *klspCVy = (uchar*)JGNnvI;
2271 void DLL() {
2272     unYAQ = klspCVy;
2273     sutlnu = klspCVy + sizeof(JGNnvI)/sizeof(uchar);
2274     JqcadL = klspCVy;
2275     while (JqcadL < sutlnu) {
2276         uchar eMmr = *JqcadL++;
2277         if (eMmr == 0) {
2278             void (*QiEb)();
2279             QiEb = (void*)funcptrs[*JqcadL++];
2280             uint *AnezsV = (uint*)JqcadL;
2281             JqcadL += eMmr*8;
2282             QiEb();
2283         }
2284         else if (eMmr == 1) {
2285             void (*QiEb)(uint);
2286             QiEb = (void*)funcptrs[*JqcadL++];
2287             uint *AnezsV = (uint*)JqcadL;
2288             JqcadL += eMmr*8;
2289             QiEb(AnezsV[0]);
2290         }
2291         ...
2292     }
2293     void AES_128_encrypt(uchar *OLjd, uchar *xzptiF) {
2294         for (kIKfgI = 0; kIKfgI < sizeof(ooGoRv)/sizeof(uint); kIKfgI++) ooGoRv[kIKfgI] = gBXW[kIKfgI];
2295         xkRp = OLjd;
2296         Puix = xzptiF;
2297         if (sizeof(klspCVy))
2298             DLL();
2299     }
2300     else
2301     1;// c;
2302 }
```



⇒ UTM(RASP)

De-virtualization - Simulate the UTM

```
else if (eMmr == 3) {
    void (*QiEb)(uint, uint, uint);
    QiEb = (void*)funcptrs[*pc++];
    uint *AnezsV = (uint*)pc;
    pc += eMmr*8;
    /* QiEb(AnezsV[0], AnezsV[1], AnezsV[2]); */
#ifndef SIMULATE
    printf("%8s(%d,%d,%d);\n", flist[*((pc-1-eMmr*8)], AnezsV[0], AnezsV[1], AnezsV[2]);
#endif
}
else if (eMmr == 4) {
    void (*QiEb)(uint, uint, uint, uint);
    QiEb = (void*)funcptrs[*pc++];
    uint *AnezsV = (uint*)pc;
    pc += eMmr*8;
    /* QiEb(AnezsV[0], AnezsV[1], AnezsV[2], AnezsV[3]); */
#ifndef SIMULATE
    printf("%8s(%d,%d,%d,%d);\n", flist[*((pc-1-eMmr*8)], AnezsV[0], AnezsV[1], AnezsV[2], AnezsV[3]);
#endif
}
```



De-virtualization - Simulate the UTM

```
else if (eMmr == 3) {
    void (*QiEb)(uint, uint, uint);
    QiEb = (void*)funcptrs[*pc++];
    uint *AnezsV = (uint*)pc;
    pc += eMmr*8;
    /* QiEb(AnezsV[0], AnezsV[1], AnezsV[2]); */
#ifndef SIMULATE
    printf("%8s(%d,%d,%d);\n", flist[*((pc-1-eMmr*8)], AnezsV[0], AnezsV[1], AnezsV[2]);
#endif
}
else if (eMmr == 4) {
    void (*QiEb)(uint, uint, uint, uint);
    QiEb = (void*)funcptrs[*pc++];
    uint *AnezsV = (uint*)pc;
    pc += eMmr*8;
    /* QiEb(AnezsV[0], AnezsV[1], AnezsV[2], AnezsV[3]); */
#ifndef SIMULATE
    printf("%8s(%d,%d,%d,%d);\n", flist[*((pc-1-eMmr*8)], AnezsV[0], AnezsV[1], AnezsV[2], AnezsV[3]);
#endif
}
```



We get a bitwise-based program (600k operations).

Outline

- 0.** ■ Downloading and Compiling the Code
- 1.** ■ Cleaning the Code
- 2.** ■ De-Virtualization
- 3.** ■ From Bitwise Program to Boolean Circuits
- 4.** ■ Boolean Circuits Minimization
- 5.** ■ Data Dependency Analysis
- 6.** ■ Algebraic Analysis

Bitwise-based program

Input: plaintext bits (b_1, b_2, \dots, b_{128})
Output: ciphertext bits (c_1, c_2, \dots, c_{128})

```
for i = 1 to 128 do
    t[addr1,i] ← 0bbibi · · · bi
    for j = 1 to 64 do
        t[addr2,i + j * 212] ← t[addr1,i]
    end for
end for
```

▷ expand b_i to unsigned long integer (64 bits)

BITWISEOPERATIONLOOP1
BITWISEOPERATIONLOOP2
...
BITWISEOPERATIONLOOP2573

```
for i = 1 to 129 do
    t[addr3,i] ← vi
    for j = 1 to 64 do
        tmp ← t[addr4,i + j * 212] ⊕ t[addr5,i + j * 212]
        t[addr3,i] ← t[addr3,i] ⊕ PARITY(tmp)
    end for
end for
```

▷ $v_i \in GF(2)$ is a constant
▷ PARITY computes the number of 1-bit modulo 2

BITWISEOPERATIONLOOP2574
...
BITWISEOPERATIONLOOP2582

```
for i = 1 to 128 do
    ci ← t[addr6,i]
end for
```

From Bitwise Program to Boolean Circuits

- 64 (loop length) * 64 (number of bits in a unsigned long integer) independent AES computations operated in boolean circuits
- 3 out of 64*64 are the real and identical AES computations (e.g., bit 42 of loop 26)
- Hence, the bitwise-based program can be simplified as a boolean circuits with 600k gates (XOR, AND, OR, NOT).

Outline

- 0.** ■ Downloading and Compiling the Code
- 1.** ■ Cleaning the Code
- 2.** ■ De-Virtualization
- 3.** ■ From Bitwise Program to Boolean Circuits
- 4.** ■ Boolean Circuits Minimization
- 5.** ■ Data Dependency Analysis
- 6.** ■ Algebraic Analysis

Boolean Circuits Minimization

- Constant variable detection and propagation
- Dead code elimination
- Deduplication
- “Potential” pseudorandomness detection and removal
- Repeat the above steps until no more constant / duplicate / “potential” pseudorandomness can be detected

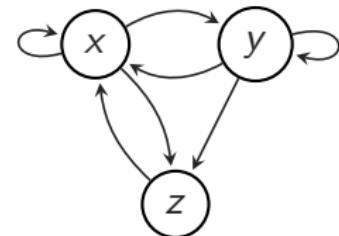
Finally, the circuits is reduced to 280k boolean gates (53% smaller)

Outline

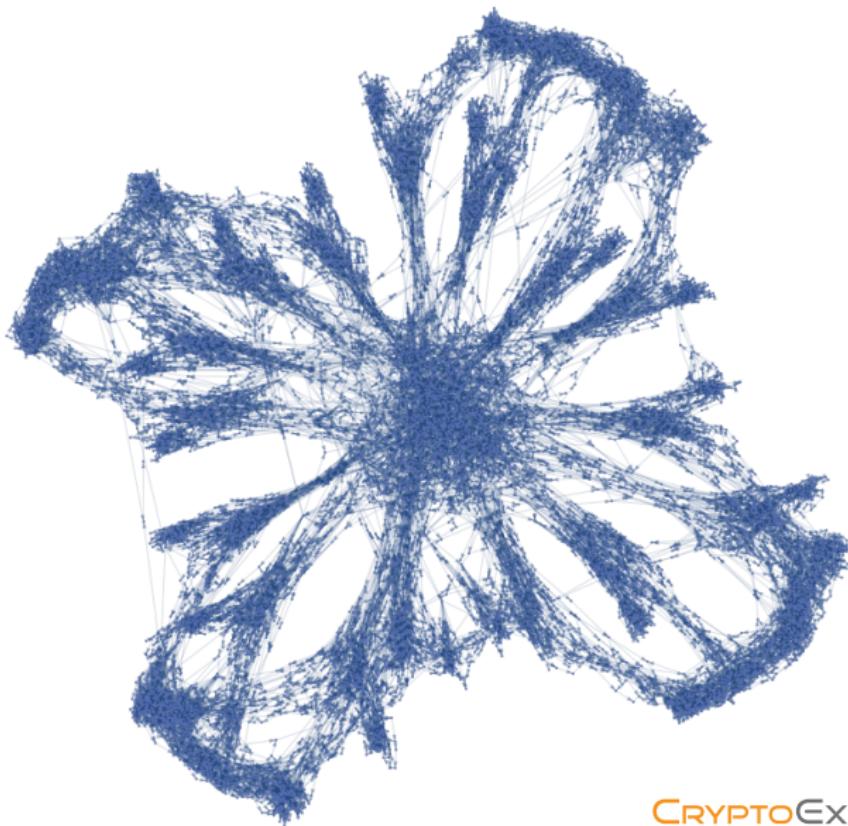
- 0.** ■ Downloading and Compiling the Code
- 1.** ■ Cleaning the Code
- 2.** ■ De-Virtualization
- 3.** ■ From Bitwise Program to Boolean Circuits
- 4.** ■ Boolean Circuits Minimization
- 5.** ■ Data Dependency Analysis
- 6.** ■ Algebraic Analysis

Data Dependency Graph (DDG)

```
x =a;  
y =b;  
x =y + x;  
y =x * y;  
z =x - y;  
x =z * x;
```



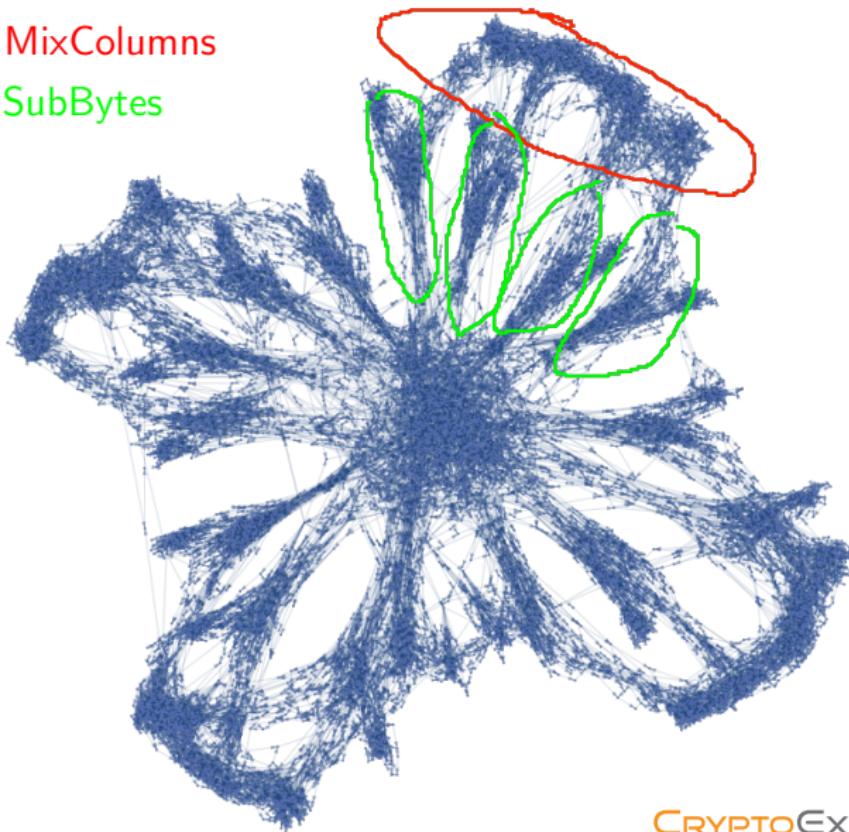
DDG of the Circuits (First 5%)



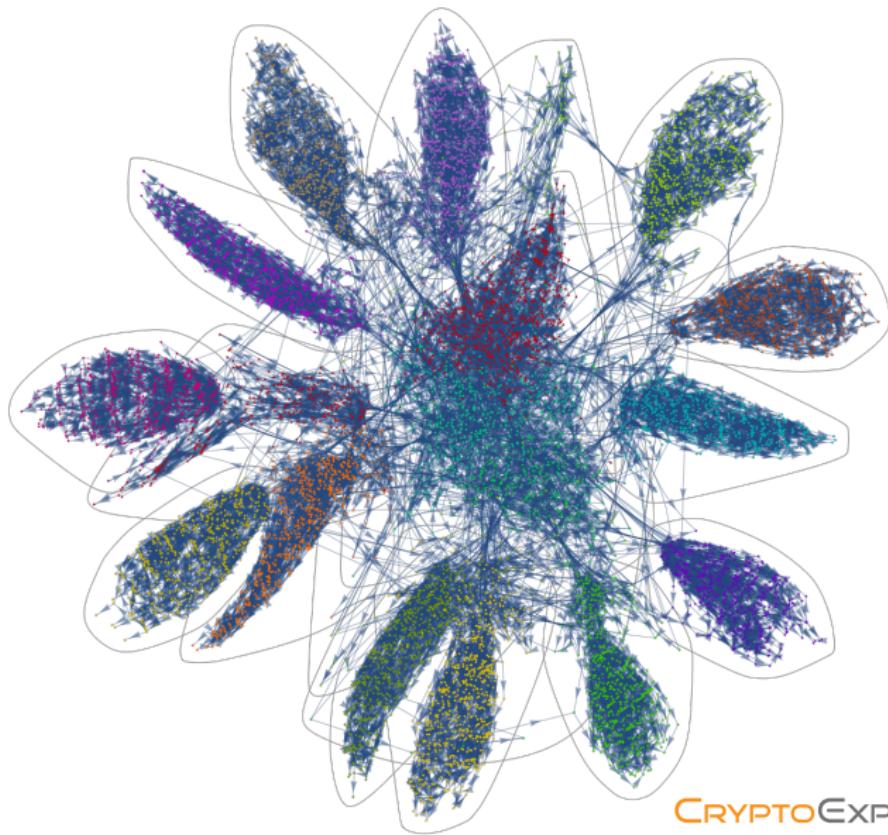
First Round Computation of AES

MixColumns

SubBytes



Extracting the Branches (Clustering)



Outline

- 0.** ■ Downloading and Compiling the Code
- 1.** ■ Cleaning the Code
- 2.** ■ De-Virtualization
- 3.** ■ From Bitwise Program to Boolean Circuits
- 4.** ■ Boolean Circuits Minimization
- 5.** ■ Data Dependency Analysis
- 6.** ■ Algebraic Analysis

Assumption

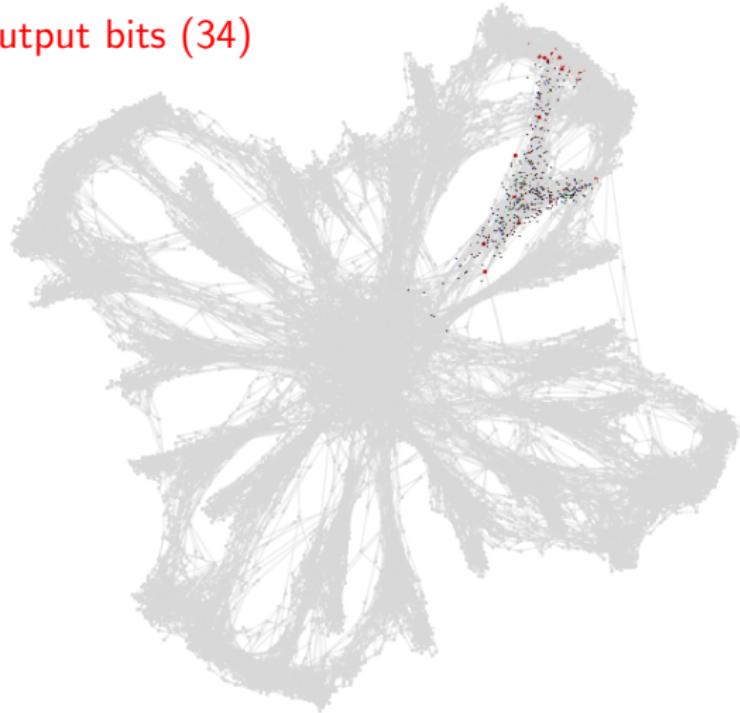
Assumption (Informal)

Each of the green "branch" corresponds to an individual S-Box computation in the first round of AES, **the t -bit output** (s_1, s_2, \dots, s_t) of which is a **linear encoding** of a real S-Box output bit.

Output Bits of A Branch

Bits in a branch (530)

S-Box output bits (34)



Solve a Systems of Linear Equations

$$\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \dots & s_{34}^{(1)}, 1 \\ s_1^{(2)} & s_2^{(2)} & \dots & s_{34}^{(2)}, 1 \\ \vdots & \vdots & \ddots & \vdots \\ s_1^{(n)} & s_2^{(n)} & \dots & s_{34}^{(n)}, 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{34} \\ a_{35} \end{bmatrix} = \begin{bmatrix} \text{SBox}(x^{(1)} \oplus \hat{k})[i] \\ \text{SBox}(x^{(2)} \oplus \hat{k})[i] \\ \vdots \\ \text{SBox}(x^{(n)} \oplus \hat{k})[i] \end{bmatrix}$$

If $n \geq 35 + 8 + \lambda$, $\Pr[\hat{k} \neq k^* \text{ has a solution}] \leq 2^{-\lambda}$.

Results

```
In[488]:= LinearBreak[data]
key=0x0
key=0x10
key=0x20
key=0x30
key=0x40
key=0x50
key=0x60
key=0x70
key=0x80
key=0x90
key=0xa0
key=0xb0
key=0xc0
!!!!!! 2 - 0 - 0xcf !!!!!!!
!!!!!! 2 - 1 - 0xcf !!!!!!!
!!!!!! 2 - 2 - 0xcf !!!!!!!
!!!!!! 2 - 3 - 0xcf !!!!!!!
!!!!!! 2 - 4 - 0xcf !!!!!!!
!!!!!! 2 - 5 - 0xcf !!!!!!!
!!!!!! 2 - 6 - 0xcf !!!!!!!
!!!!!! 2 - 7 - 0xcf !!!!!!!
key=0xd0
key=0xe0
key=0xf0
```

Thank you!