

CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2025 Lecture 16

Announcements

1. HW 7 is due next lecture!
2. HW 8 will be available on Tuesday.

Last time:

- Random number generation.

This time:

- Index calculus and the number field sieve

Factoring algorithms we've seen already

- Trial division
- Pollard rho
- Pollard $p-1$
- Elliptic curve method

Discrete log algorithms we've seen already

- Brute force
- Baby step giant step
- Pollard rho
- Pohlig-Hellman

Fermat factorization

Input: Integer N to be factored

Write $N = a^2 - b^2 = (a + b)(a - b)$.

Then if $N = uv = a^2 - b^2$ for $a = \frac{1}{2}(u + v)$ and $b = \frac{1}{2}(u - v)$.

Algorithm

For $\lceil \sqrt{N} \rceil \leq a \leq (N + a)/6$:

 If $b = \sqrt{a^2 - N} \in \mathbb{Z}$:

 return $a - b$

Fermat factorization

Input: Integer N to be factored

Write $N = a^2 - b^2 = (a + b)(a - b)$.

Then if $N = uv = a^2 - b^2$ for $a = \frac{1}{2}(u + v)$ and $b = \frac{1}{2}(u - v)$.

Algorithm

For $\lceil \sqrt{N} \rceil \leq a \leq (N + a)/6$:

 If $b = \sqrt{a^2 - N} \in \mathbb{Z}$:

 return $a - b$

Up to p steps to find a factor p .

This has $O(\sqrt{N})$ running time, not competitive with other factoring algorithms we've seen already.

Quadratic Sieve Intuition

Main insight: Try to construct two squares a^2 and b^2 such that

$$a^2 \equiv b^2 \pmod{N}$$

Then

$$a^2 - b^2 = (a + b)(a - b) \equiv 0 \pmod{N}$$

and hope that one of $a + b$ or $a - b$ shares a nontrivial common factor with N .

Quadratic Sieve Intuition

Main insight: Try to construct two squares a^2 and b^2 such that

$$a^2 \equiv b^2 \pmod{N}$$

Then

$$a^2 - b^2 = (a + b)(a - b) \equiv 0 \pmod{N}$$

and hope that one of $a + b$ or $a - b$ shares a nontrivial common factor with N .

Fermat factorization iterates through options for $\sqrt{a^2 - N}$ until it finds a perfect square.

With more cleverness, we can *construct* a square rather than searching for it.

Quadratic Sieve

Intuition

We might not find a square outright, but we can construct a square as a product of numbers we test.

Quadratic Sieve

Intuition

We might not find a square outright, but we can construct a square as a product of numbers we test.

1. **Relation-finding** Start at $c = \lceil \sqrt{N} \rceil$.
2. Try to factor each of $c^2 - N, (c + 1)^2 - N, \dots$
3. Only save a $d_i = c_i^2 - N$ if all of its prime factors are less than some bound B . (If it is *B-smooth*.)
4. Store each d_i by its exponent vector

$$d_i = 2^{e_2} 3^{e_3} \dots B^{e_B}$$

5. If $\prod_i d_i$ is a square, then its exponent vector contains only even entries.

Quadratic Sieve

1. **Relation-finding** Save B -smooth values $d_i = c_i^2 - N$.
2. Store each d_i by its exponent vector $d_i = 2^{e_2} 3^{e_3} \dots B^{e_B}$.
3. If $\prod_i d_i$ is a square, then its exponent vector contains only even entries.

Quadratic Sieve

1. **Relation-finding** Save B -smooth values $d_i = c_i^2 - N$.
2. Store each d_i by its exponent vector $d_i = 2^{e_2} 3^{e_3} \dots B^{e_B}$.
3. If $\prod_i d_i$ is a square, then its exponent vector contains only even entries.

Let $p_B = \# \text{of primes} \leq B$.

4. **Linear Algebra** Once $> p_B$ factorizations have been collected, find a linear dependence mod 2 in the exponent vectors, so

$$\prod_{i \in S} d_i \text{ is a square}$$

Quadratic Sieve

1. **Relation-finding** Save B -smooth values $d_i = c_i^2 - N$.
2. Store each d_i by its exponent vector $d_i = 2^{e_2} 3^{e_3} \dots B^{e_B}$.
3. If $\prod_i d_i$ is a square, then its exponent vector contains only even entries.
Let $p_B = \# \text{of primes} \leq B$.
4. **Linear Algebra** Once $> p_B$ factorizations have been collected, find a linear dependence mod 2 in the exponent vectors, so

$$\prod_{i \in S} d_i \text{ is a square}$$

5. **Square roots** Compute square root $\sqrt{\prod_{i \in S} d_i}$ and hope for a nontrivial factorization of N .
6. Square product has 50% chance of factoring pq .

An example of the quadratic sieve

Let's try to factor $N = 2759$.

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

$$55^2 - 2759 = 266.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

$$55^2 - 2759 = 266.$$

$$56^2 - 2759 = 377.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

$$55^2 - 2759 = 266.$$

$$56^2 - 2759 = 377.$$

$$57^2 - 2759 = 490 = 2 \cdot 5 \cdot 7^2.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

$$55^2 - 2759 = 266.$$

$$56^2 - 2759 = 377.$$

$$57^2 - 2759 = 490 = 2 \cdot 5 \cdot 7^2.$$

$$58^2 - 2759 = 605 = 5 \cdot 11^2.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

$$55^2 - 2759 = 266.$$

$$56^2 - 2759 = 377.$$

$$57^2 - 2759 = 490 = 2 \cdot 5 \cdot 7^2.$$

$$58^2 - 2759 = 605 = 5 \cdot 11^2.$$

Linear Algebra: The *product* $50 \cdot 490 \cdot 605$ is a square:

$$2^2 \cdot 5^4 \cdot 7^2 \cdot 11^2.$$

An example of the quadratic sieve

Let's try to factor $N = 2759$.

Sieving values $(\lceil \sqrt{N} + i \rceil)^2 \bmod N$:

$$53^2 - 2759 = 50 = 2 \cdot 5^2.$$

$$54^2 - 2759 = 157.$$

$$55^2 - 2759 = 266.$$

$$56^2 - 2759 = 377.$$

$$57^2 - 2759 = 490 = 2 \cdot 5 \cdot 7^2.$$

$$58^2 - 2759 = 605 = 5 \cdot 11^2.$$

Linear Algebra: The *product* $50 \cdot 490 \cdot 605$ is a square:

$$2^2 \cdot 5^4 \cdot 7^2 \cdot 11^2.$$

Recall idea: If $a^2 - N$ is a square b^2 then $N = (a - b)(a + b)$.

QS computes $\gcd\{2759, 53 \cdot 57 \cdot 58 - \sqrt{50 \cdot 490 \cdot 605}\} = 31$.

Quadratic Sieve running time

- Let B be the maximum prime factor allowed: “smoothness bound”
- How do we choose B ?
- How many numbers do we have to try to factor?
- Depends on (heuristic) probability that a randomly chosen number is B -smooth.

$$\Pr(x \text{ is } B\text{-smooth}) \approx u^{-u} \quad u = \frac{\ln x}{\ln B}$$

Quadratic Sieve running time

- Let B be the maximum prime factor allowed: “smoothness bound”
- How do we choose B ?
- How many numbers do we have to try to factor?
- Depends on (heuristic) probability that a randomly chosen number is B -smooth.

$$\Pr(x \text{ is } B\text{-smooth}) \approx u^{-u} \quad u = \frac{\ln x}{\ln B}$$

- Sieving costs $\ln \ln B$ operations per number.
- Expected values to sieve per B -smooth relation: u^u .
- $\#$ primes $\leq B = \#$ of relations needed $= B / \ln B$

Quadratic Sieve running time

- Let B be the maximum prime factor allowed: “smoothness bound”
- How do we choose B ?
- How many numbers do we have to try to factor?
- Depends on (heuristic) probability that a randomly chosen number is B -smooth.

$$\Pr(x \text{ is } B\text{-smooth}) \approx u^{-u} \quad u = \frac{\ln x}{\ln B}$$

- Sieving costs $\ln \ln B$ operations per number.
- Expected values to sieve per B -smooth relation: u^u .
- # primes $\leq B =$ # of relations needed $= B / \ln B$

Running time: $L_N(1/2, 1) = e^{(1+o(1))\sqrt{\ln N \ln \ln N}}$.

L-notation:

$$L_n(\alpha, c) = e^{(c+o(1))(\ln n)^\alpha (\ln \ln n)^{1-\alpha}}$$

Number field sieve

Best running time for general purpose factoring

Insight

Instead of using homomorphism $\mathbb{Z} \rightarrow \mathbb{Z}/N\mathbb{Z}$
use ring homomorphism $\mathbb{Z}/f(x) \rightarrow \mathbb{Z}/N\mathbb{Z}$

Algorithm

1. **Polynomial selection** Find a good choice of number field K .
2. **Relation finding** Factor elements over \mathcal{O}_K and over \mathbb{Z} .
3. **Linear algebra** Find a square in \mathcal{O}_K and a square in \mathbb{Z}
4. **Square roots** Take square roots, map into \mathbb{Z} , and hope we find a factor.

Polynomial selection details

Pick $m \in \mathbb{Z}$, $m \approx N^{1/6}$,

$$f(m) = n_0 + n_1m + \cdots + n_5m^5 + m^6 = f(m) = N.$$

For root $\alpha \in \mathbb{C}$, $f(\alpha) = 0$:

Defines homomorphism $\phi : \alpha \mapsto m$.

$$\phi(a_0 + a_1\alpha + \cdots + a_5\alpha^5) = a_0 + a_1m + \cdots + a_5m^5$$

Desirable properties:

- f has small coefficients.
- f has favorable heuristics for lots of evaluation points that are smooth

Relation finding details

We are aiming to find a square in \mathbb{Z} and a square in $\mathbb{Z}[\alpha]$.

Sieve over pairs $a, b \in \mathbb{Z}$ to find pairs where

$$a - bm \text{ } B\text{-smooth} \quad N(a - b\alpha) \text{ } B\text{-smooth}$$

$$N(a - b\alpha) = \prod_i (a - b\alpha_i) = b^d \prod_i (a/b - \alpha_i) = b^d f(a/b)$$

x is a square $\implies N(x)$ square. Not sufficient but you can work around this.

Traditionally implemented using sieving, hence the name.

Now state of the art uses sieving, ECM, and batch factorization methods to find relations.

Embarrassingly parallel. Occupies most of the computation time.

Linear algebra details

Use linear algebra to find vectors summing to 0 mod 2 on both sides.

Matrix is a sparse matrix mod 2.

Use specialized algorithms like block Wiedemann algorithm:
 $O(wM^2)$ for a $M \times M$ matrix with row weight w .

Sample parameters: 282M rows, weight 200.

Parallelizes, but not embarrassingly. Second most expensive phase after sieving.

Square root

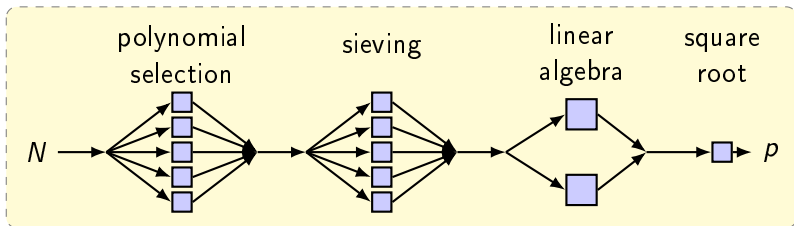
Once a square is found, need to take square roots:

$$\prod_S (a - b\alpha) = \gamma^2 \in \mathbb{Z}[\alpha]$$

$$\prod_S (a - bm) = v^2 \in \mathbb{Z}$$

Try to factor N as $(\phi(\gamma) + v)(\phi(\gamma) - v)$.

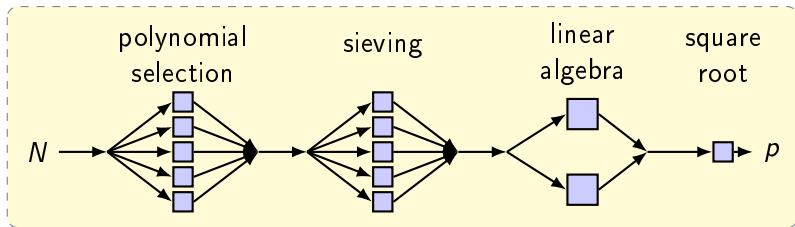
How long does factoring take with the number field sieve?



Answer 1

$$L_N(1/3, \sqrt[3]{64/9}) = e^{(1.923+o(1))(\ln p)^{1/3}(\ln \ln p)^{2/3}}$$

How long does factoring take with the number field sieve?



Answer 2

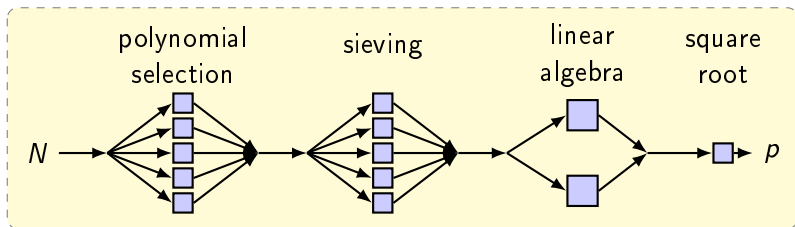
512-bit RSA: < 1 core-year

768-bit RSA: $< 1,000$ core-years

1024-bit RSA: $\approx 500,000$ core-years

2048-bit RSA: Minimum recommended key size today.

How long does factoring take with the number field sieve?



Answer 3

512-bit RSA: 7 months; large academic effort [CBLMMtRZ 1999]

768-bit RSA: 3 years; large academic effort [KAFLTBGKMOtRTZ 2009]

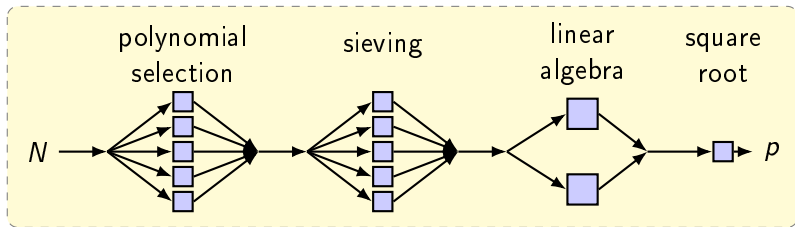
795-bit RSA: 6 months; academic effort [BGGHTZ 2019]

829-bit RSA: 3 months; academic effort [BGGHTZ 2020]

512-bit RSA: 2.5 months — single machine [Moody 2009]

512-bit RSA: < 4 hours — Amazon EC2 cluster [VCLFBH 2016]

How long does factoring take with the number field sieve?



	polysel	sieving	linalg	sqrt
795-bit RSA	76 core-years	794 core-years	83 core-years	17 hours

512-bit RSA in the wild.

International Traffic in Arms Regulations

April 1, 1992 version

Category XIII--Auxiliary Military Equipment ...

(b) Information Security Systems and equipment, cryptographic devices, software, and components specifically designed or modified therefore, including:

(1) Cryptographic (including key management) systems, equipment, assemblies, modules, integrated circuits, components or software with the capability of maintaining secrecy or confidentiality of information or information systems, except cryptographic equipment and software as follows:

(i) Restricted to decryption functions specifically designed to allow the execution of copy protected software, provided the decryption functions are not user-accessible.

(ii) Specially designed, developed or modified for use in machines for banking or money transactions, and restricted to use only in such transactions. Machines for banking or money transactions include automatic teller machines, self-service statement printers, point of sale terminals or equipment for the encryption of interbanking transactions.

...

How do you selectively weaken a protocol based on RSA?

How do you selectively weaken a protocol based on RSA?

Export answer: Optionally use a small RSA key.

Timeline of US cryptography export control

- Pre-1994: Encryption software requires individual export license as a munition.
- 1994: US State Department amends ITAR regulations to allow export of approved software to approved countries without individual licenses. 40-bit symmetric cryptography was understood to be approved under this scheme.
- 1995: Netscape develops initial SSL protocol.
- 1996: Bernstein v. United States; California judge rules ITAR regulations are unconstitutional because “code is speech”
- 1996: Cryptography regulation moved to Department of Commerce.
- 1999: TLS 1.0 standardized.
- 2000: Department of Commerce loosens regulations on mass-market and open source software.

Commerce Control List: Category 5 - Info. Security

(current)

a.1.a. A symmetric algorithm employing a key length in excess of 56-bits; or

a.1.b. An asymmetric algorithm where the security of the algorithm is based on any of the following:

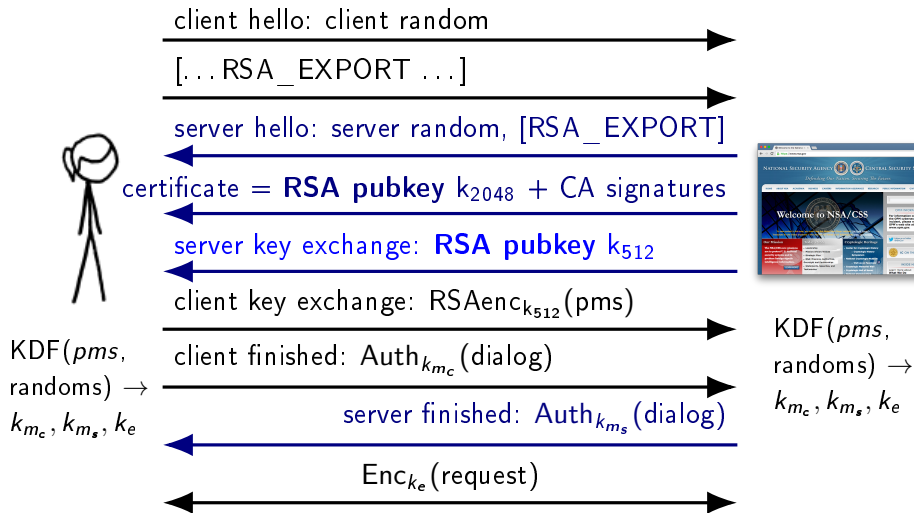
a.1.b.1. Factorization of integers in excess of 512 bits (e.g., RSA);

a.1.b.2. Computation of discrete logarithms in a multiplicative group of a finite field of size greater than 512 bits (e.g., Diffie-Hellman over Z/pZ); or

a.1.b.3. Discrete logarithms in a group other than mentioned in 5A002.a.1.b.2 in excess of 112 bits (e.g., Diffie-Hellman over an elliptic curve);

a.2. Designed or modified to perform cryptanalytic functions;

TLS RSA Export Key Exchange



RSA export cipher suites in TLS

In March 2015, export cipher suites supported by 36.7% of the 14 million sites serving browser-trusted certificates!

```
TLS_RSA_EXPORT_WITH_RC4_40_MD5
```

```
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
```

```
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
```

Totally insecure, but no modern client should negotiate export ciphers.

FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]

client hello: random

[... RSA ...]



FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]

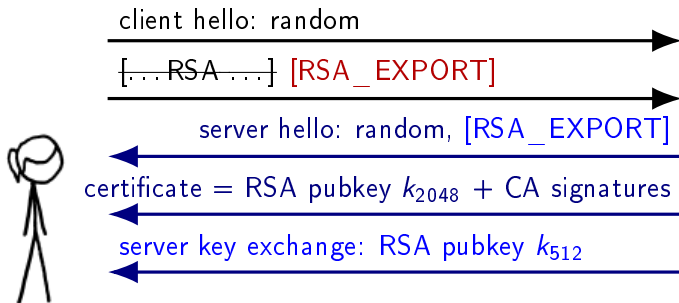
client hello: random

[...RSA...] [RSA_EXPORT]



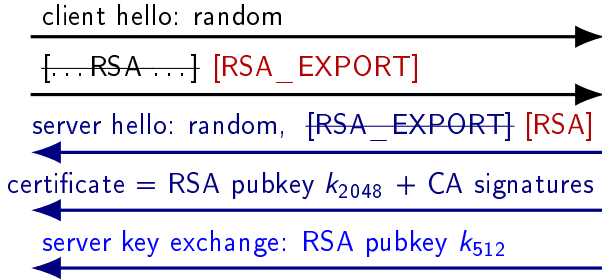
FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



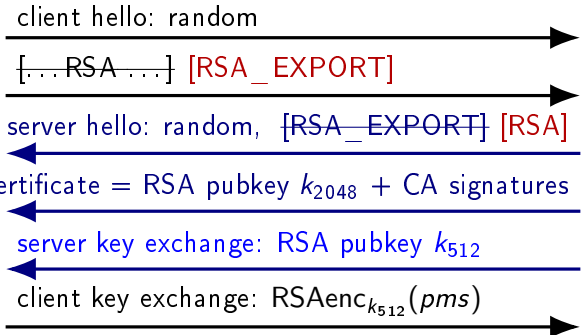
FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



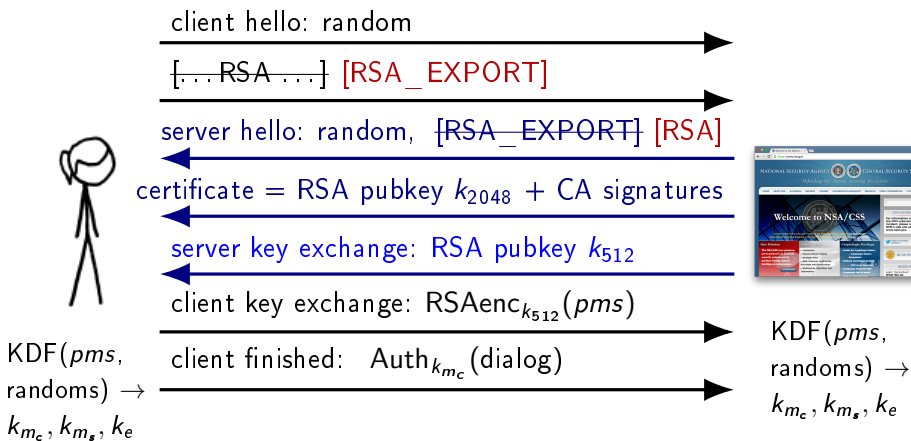
$\text{KDF}(pms, \text{randoms}) \rightarrow k_{m_c}, k_{m_s}, k_e$



$\text{KDF}(pms, \text{randoms}) \rightarrow k_{m_c}, k_{m_s}, k_e$

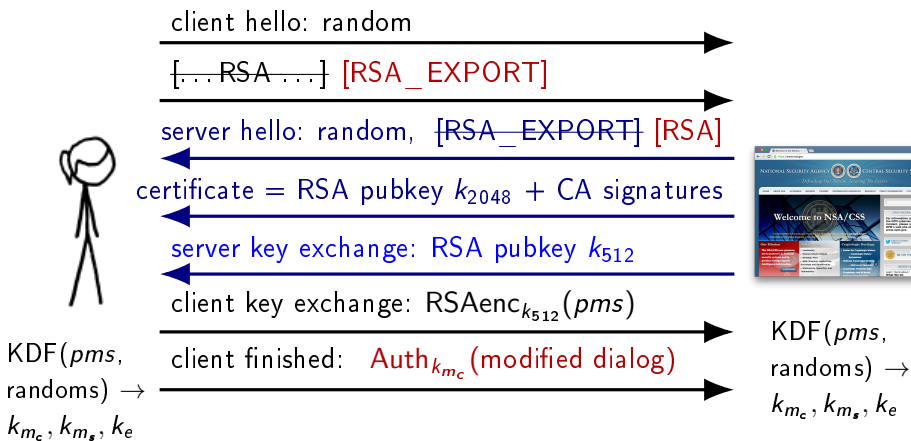
FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



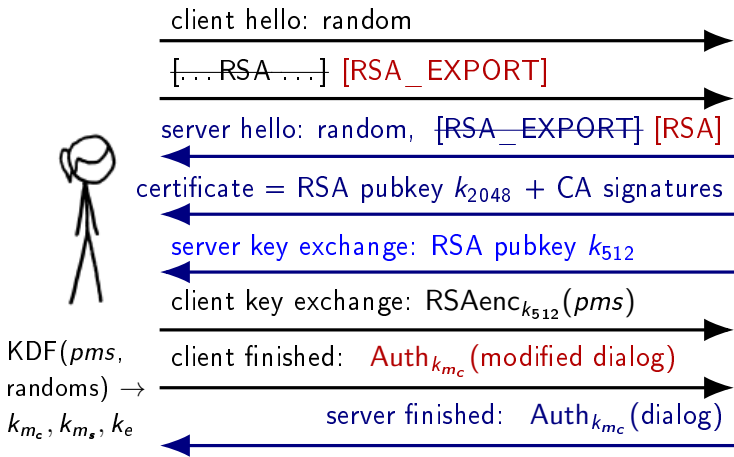
FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



FREAK: MITM downgrade attack to export RSA

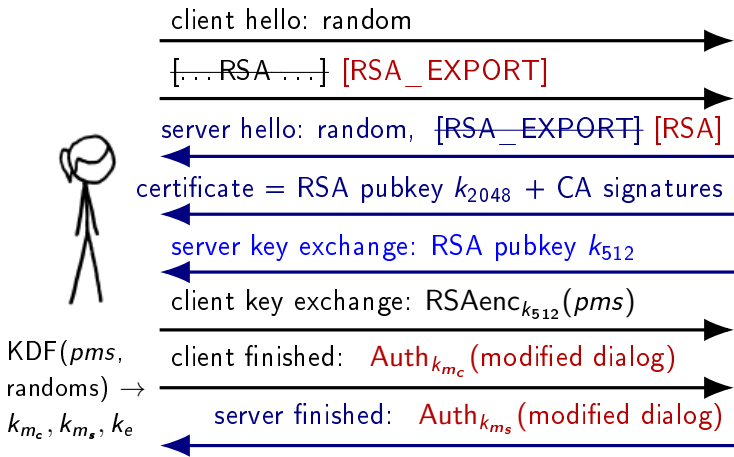
Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



$KDF(pms, \text{randoms}) \rightarrow k_{m_c}, k_{m_s}, k_e$

FREAK: MITM downgrade attack to export RSA

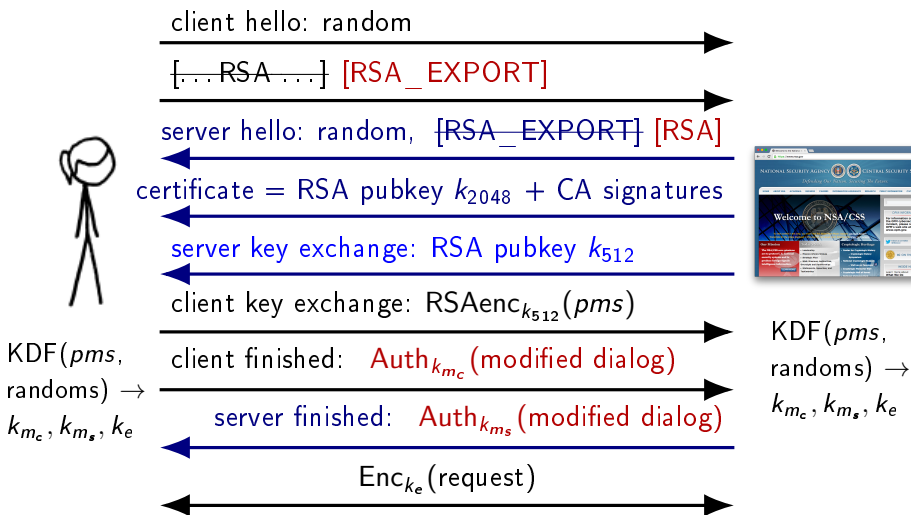
Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



$KDF(pms, \text{randoms}) \rightarrow k_{m_c}, k_{m_s}, k_e$

FREAK: MITM downgrade attack to export RSA

Implementation flaw: Most major browsers accepted unexpected server key exchange messages. [BDFKPSZZ 2015]



FREAK vulnerability in practice

- Implementation flaw affected OpenSSL, Microsoft SChannel, IBM JSSE, Safari, Android, Chrome, BlackBerry, Opera, IE
- Attack outline:
 1. MITM attacker downgrades connection to export, learns server's ephemeral 512-bit RSA export key.
 2. Attacker factors 512-bit modulus to obtain server private key.
 3. Attacker uses private key to forge client/server authentication for successful downgrade.
- Attacker challenge: Need to know 512-bit private key before connection times out
- Implementation shortcut: "Ephemeral" 512-bit RSA server keys generated only on application start; last for hours, days, weeks, months.

“Perfect Forward Secrecy”

Common incorrect beliefs in the security community in 2015

“Sites that use perfect forward secrecy can provide better security to users in cases where the encrypted data is being monitored and recorded by a third party.”

“With Perfect Forward Secrecy, anyone possessing the private key and a wiretap of Internet activity can decrypt nothing.”

“Ideally the DH group would match or exceed the RSA key size but 1024-bit DHE is arguably better than straight 2048-bit RSA so you can get away with that if you want to.”

“But in practical terms the risk of private key theft, for a non-ephemeral key, dwarfs out any cryptanalytic risk for any RSA or DH of 1024 bits or more; in that sense, PFS is a must-have and DHE with a 1024-bit DH key is much safer than RSA-based cipher suites, regardless of the RSA key size.”

Index calculus for discrete log

Want to solve $g^x = y \pmod p$, $\text{ord}(g) = q$.

1. **Relation finding** Search for smooth elements

$$g^{x_i} = p_1^{r_{i1}} \dots p_B^{r_{iB}} \pmod p$$

Index calculus for discrete log

Want to solve $g^x = y \pmod p$, $\text{ord}(g) = q$.

1. **Relation finding** Search for smooth elements

$$g^{x_i} = p_1^{r_{i1}} \dots p_B^{r_{iB}} \pmod p$$

2. **Linear algebra** Once we have $\geq B$ relations:

$$\begin{bmatrix} x_1 & \equiv & r_{11} \log_g p_1 + \dots + r_{1B} \log_g p_B \pmod q \\ \vdots & & \\ x_k & \equiv & r_{k1} \log_g p_1 + \dots + r_{kB} \log_g p_B \pmod q \end{bmatrix}$$

use linear algebra to solve for $\log_g p_i$ for small p_i :

$$\log_g p_i = s_j x_1 + \dots + s_{ik} x_k \pmod q$$

Index calculus for discrete log

Want to solve $g^x = y \pmod p$, $\text{ord}(g) = q$.

1. **Relation finding** Search for smooth elements

$$g^{x_i} = p_1^{r_{i1}} \dots p_B^{r_{iB}} \pmod p$$

2. **Linear algebra** Once we have $\geq B$ relations:

$$\begin{bmatrix} x_1 & \equiv & r_{11} \log_g p_1 + \dots + r_{1B} \log_g p_B \pmod q \\ \vdots & & \\ x_k & \equiv & r_{k1} \log_g p_1 + \dots + r_{kB} \log_g p_B \pmod q \end{bmatrix}$$

use linear algebra to solve for $\log_g p_i$ for small p_i :

$$\log_g p_i = s_j x_1 + \dots + s_{ik} x_k \pmod q$$

3. **Individual log** Search for $g^R y$ smooth mod p :

$$g^R y \equiv p_1^{t_1} \dots p_B^{t_B} \pmod p$$

and solve: $R + \log_g y = t_1 \log_g p_1 + \dots + t_B \log_g p_B \pmod q$

Index calculus running time

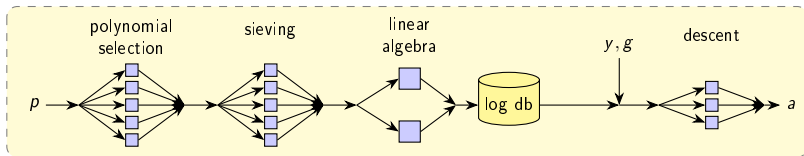
Use same probability estimates as in factoring.

Optimize running time by balancing relation finding, linear algebra, and individual log phases.

$$L_p(1/2, \sqrt{2}) = e^{(\sqrt{2}+o(1))\sqrt{\ln p \ln \ln p}}$$

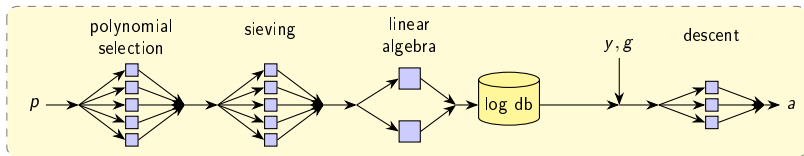
Number field sieve discrete log algorithm

[Gordon], [Joux, Lercier], [Semaev]



1. **Polynomial selection:** Choose polynomial/number field K .
2. **Relation collection:** Factor elements over \mathcal{O}_K and over \mathbb{Z} .
3. **Linear algebra** Once there are enough relations, solve for logs of small elements.
4. **Individual log “Descent”** Try to write target t as sum of logs in known database.

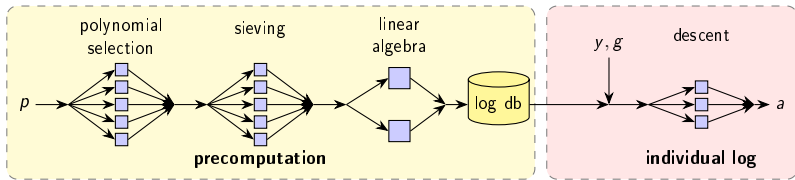
How long does it take to compute discrete logs?



Answer 1:

$$L(1/3, 1.923) = \exp(1.923(\log p)^{1/3}(\log \log p)^{2/3})$$

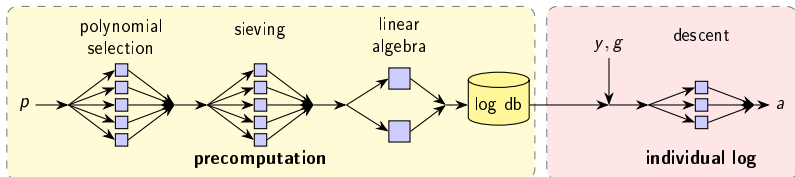
How long does it take to compute discrete logs?



Answer 1:

$$L(1/3, 1.923) = \exp(1.923(\log p)^{1/3}(\log \log p)^{2/3})$$

How long does it take to compute discrete logs?

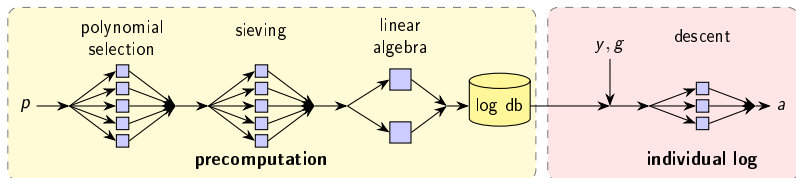


Answer 1:

$$L(1/3, 1.232)$$

$$L(1/3, 1.923) = \exp(1.923(\log p)^{1/3}(\log \log p)^{2/3})$$

How long does it take to compute discrete logs?



Answer 1:

$$L(1/3, 1.232)$$

$$L(1/3, 1.923) = \exp(1.923(\log p)^{1/3}(\log \log p)^{2/3})$$

Answer 2:

In 2005, 431-bit discrete log in 3 weeks. [Joux, Lercier]

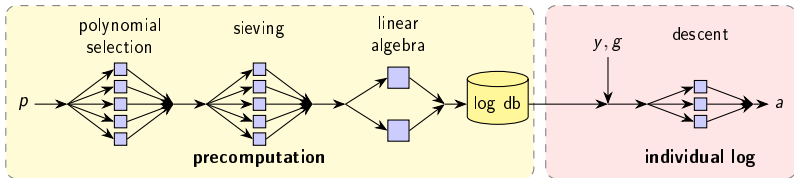
In 2007, 530-bit discrete log. [Kleinjung]

In 2014, 596-bit discrete log. [Bouvier et al.]

In 2017, 768-bit discrete log. [Kleinjung et al.]

In 2019, 795-bit discrete log. [Boudot et al.]

How long does it take to compute discrete logs?



Answer 3:

	polysel	sieving	linalg	descent
795-bit DL	152 core-years	2400 core-years	625 core-years	<1 core-year

Precomputation can be done once and reused for many individual logs!

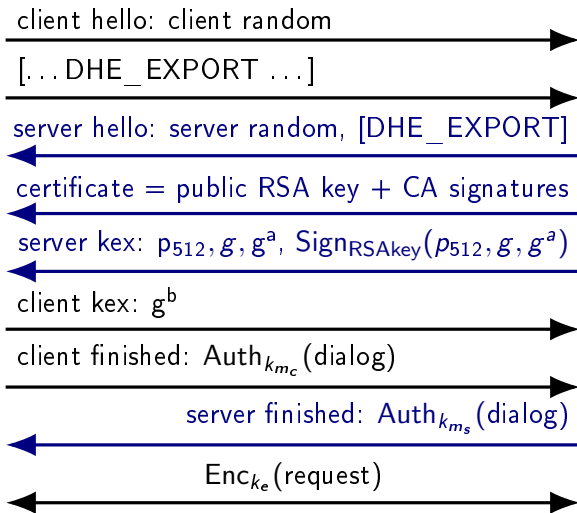
How do you selectively weaken a protocol based on Diffie-Hellman?

Export answer: Optionally use a smaller prime.

TLS Diffie-Hellman Export Key Exchange



$KDF(g^{ab},$
randoms) \rightarrow
 k_{m_c}, k_{m_s}, k_e



$KDF(g^{ab},$
randoms) \rightarrow
 k_{m_c}, k_{m_s}, k_e

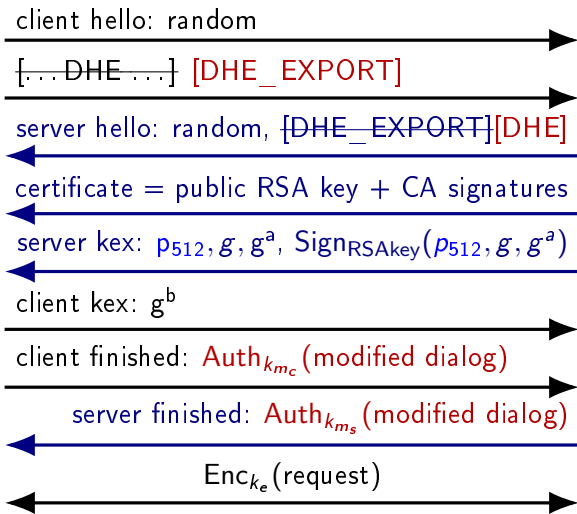
Diffie-Hellman export cipher suites in TLS

```
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA  
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA  
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA  
TLS_DH_Annon_EXPORT_WITH_RC4_40_MD5  
TLS_DH_Annon_EXPORT_WITH_DES40_CBC_SHA
```

April 2015: 8.4% of Alexa top 1M HTTPS support DHE_EXPORT.

Logjam: Active downgrade attack to export Diffie-Hellman

Protocol flaw: Server does not sign chosen cipher suite.



$\text{KDF}(g^{ab}, \text{randoms}) \rightarrow k_{m_c}, k_{m_s}, k_e$

$\text{KDF}(g^{ab}, \text{randoms}) \rightarrow k_{m_c}, k_{m_s}, k_e$

Carrying out the Diffie-Hellman export downgrade attack

1. Attacker man-in-the-middle connection, changing messages as necessary.
2. Attacker computes discrete log of g^a or g^b to learn session keys.
3. Attacker uses session keys to forge client, server finished messages.
 - Attacker challenge: compute client or server ephemeral Diffie-Hellman secrets before connection times out
 - For export Diffie-Hellman, most servers actually generate per-connection secrets.

Implementing Logjam

Parameters hard-coded in implementations or built into standards.

97% of DHE_EXPORT hosts choose one of three 512-bit primes.

Hosts	Source	Year	Bits
80%	Apache 2.2	2005	512
13%	mod_ssl 2.3.0	1999	512
4%	JDK	2003	512

- Carried out precomputation for common primes.
- After 1 week precomputation, median individual log time 70s.
- Logjam and our precomputations can be used to break connections to 8% of the HTTPS top 1M sites!

Is breaking 1024-bit Diffie-Hellman within reach of governments?

	Sieving	Linear Algebra	Descent
	core-years	core-years	core-time
RSA-512	0.5	0.33	
DH-512	2.5	7.7	10 mins
RSA-795	800	100	
DH-795	2400	600	≈ days
RSA-1024	500,000		
DH-1024	1,700,000		≈ days

Is breaking 1024-bit Diffie-Hellman within reach of governments?

	Sieving	Linear Algebra	Descent
	core-years	core-years	core-time
RSA-512	0.5	0.33	
DH-512	2.5	7.7	10 mins
RSA-795	800	100	
DH-795	2400	600	\approx days
RSA-1024	500,000		
DH-1024	1,700,000		\approx days

- Precomputation for 1024-bit p is feasible for large organizations/with special purpose hardware.
- Individual logs can be computed in close to real time

James Bamford, 2012, Wired

According to another top official also involved with the program, **the NSA made an enormous breakthrough several years ago** in its ability to cryptanalyze, or break, unfathomably complex encryption systems employed by not only governments around the world but also many average computer users in the US. The upshot, according to this official: **“Everybody’s a target; everybody with communication is a target.”**

[...]

The breakthrough was enormous, says the former official, and soon afterward the agency pulled the shade down tight on the project, even within the intelligence community and Congress. “Only the chairman and vice chairman and the two staff directors of each intelligence committee were told about it,” he says. The reason? **“They were thinking that this computing breakthrough was going to give them the ability to crack current public encryption.”**

2013 NSA "Black Budget"

"Also, we are investing in groundbreaking cryptanalytic capabilities to defeat adversarial cryptography and exploit internet traffic."

This Exhibit is SECRET//NOFORN

Program	Expenditure Center	Project	FY 2011	FY 2012	FY 2013	FY 2012 - FY 2013 Change
	Computer Network Operations	Data Acquisition and Cover Support	56,949	100,987	117,605	16,618
		GENIE	615,177	636,175	651,743	15,568
		SIGINT Enabling	298,613	275,376	254,943	-20,433
	Computer Network Operations Total		970,739	1,012,538	1,024,291	11,753
	Cryptanalysis & Exploitation Services	Analysis of Target Systems	39,429	35,128	34,321	-807
		Cryptanalytic IT Systems	130,012	136,797	247,121	110,324
		Cyber Cryptanalysis	181,834	110,673	115,300	4,627
		Exploitation Solutions	90,024	59,915	58,308	-1,607
		Microelectronics	64,603	61,672	45,886	-15,786

*numbers in thousands

Parameter reuse for 1024-bit Diffie-Hellman

- Precomputation for a single 1024-bit prime would have allowed passive decryption of connections to 66% of VPN servers and 26% of SSH servers in 2015.

(Oakley Group 2)

- Precomputation for a second common 1024-bit prime allowed passive decryption for 18% of top 1M HTTPS domains.

(Apache 2.2)



TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL



4. Communicate Results

Can we decrypt the VPN traffic?

- If the answer is “No” then explain how to turn it into a “YES!”
- If the answer is “YES!” then...

TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL



TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL

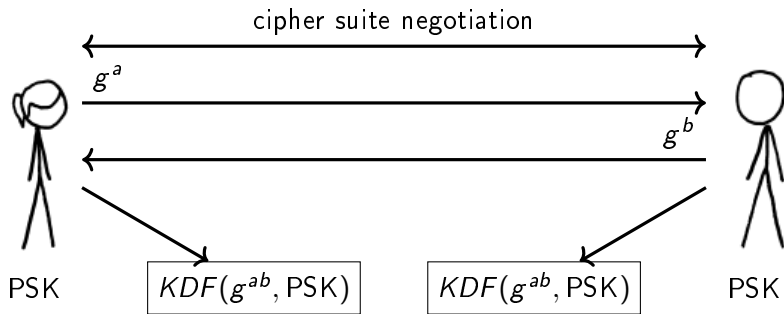
Happy Dance!!



TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL

IKE Key Exchange for IPsec VPNs

IKE chooses Diffie-Hellman parameters from standardized set.





TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL

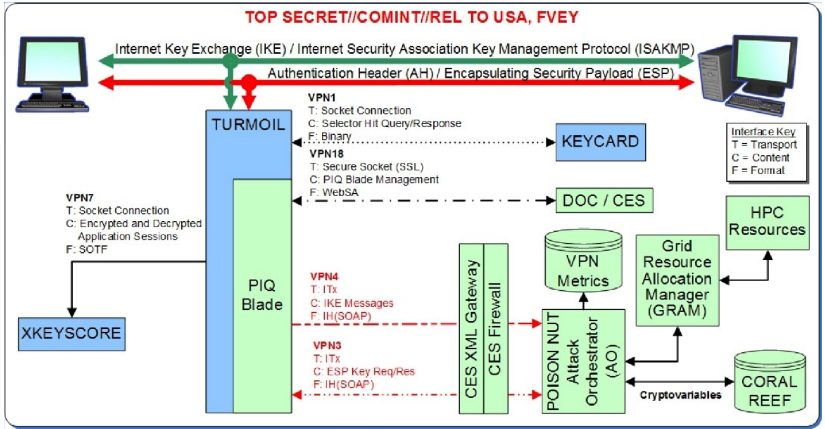
Turn that Frown Upside Down! From "No" to "YES!"



- Depends on why we couldn't decrypt it
- Find Pre-Shared Key
- Locate complete paired collect
- Locate both IKE and ESP traffic
- Have collection sites do surveys for the IP's
- Find better quality collect with rich metadata

TOP SECRET//COMINT//REL USA, AUS, CAN, GBR, NZL

NSA VPN Attack Orchestration



NSA's on-demand IKE decryption requires:

- Known pre-shared key.
- Both sides of IKE handshake.
- Both IKE handshake and ESP traffic.
- IKE+ESP data is sent to HPC resources.

Discrete log decryption would require:

- Known pre-shared key.
- Both sides of IKE handshake.
- Both IKE handshake and ESP traffic.
- IKE data sent to HPC resources.

Vulnerable servers, if attacker can precompute for

	all 512-bit p	one 1024-bit p	ten 1024-bit p
HTTPS Top 1M MITM	45K (8.4%)	205K (37.1%)	309K (56.1%)
HTTPS Top 1M	118 (0.0%)	98.5K (17.9%)	132K (24.0%)
HTTPS Trusted MITM	489K (3.4%)	1.84M (12.8%)	3.41M (23.8%)
HTTPS Trusted	1K (0.0%)	939K (6.56%)	1.43M (10.0%)
IKEv1 IPv4	–	1.69M (66.1%)	1.69M (66.1%)
IKEv2 IPv4	–	726K (63.9%)	726K (63.9%)
SSH IPv4	–	3.6M (25.7%)	3.6M (25.7%)

Diffie-Hellman Attacks and Mitigations

Logjam attack:

Mitigations:

- Major browsers raised minimum DH lengths.
- OpenSSH removed 1024-bit DH group.
- TLS 1.3 includes anti-downgrade mechanism.
- Recommendation: Don't backdoor crypto!

Mass surveillance:

Mitigations:

- Elliptic curve cryptography is safer than factoring or prime-field discrete log.
- If ECC isn't an option, use ≥ 2048 -bit primes.