CR10: Cryptanalysis Autumn 2018

Lecture 10 — December, 11^{th} 2018

Lecturers ***G. Hanrot****, E. Kirshanova* *Scribe: P. Meyer*

1 Integer Factoring Algorithms – Continued

What is the **amortised** cost of factoring a set of ($x_i^2 \mod N$) where the x_i are of the form $x_i = \prod_{j=1}^{r} p_j^{u_{ij}}$? A case of special interest is if additionally $x_i = i + \sqrt{N}$, which we will assume in the following. Note that in that case $x_i^2 \mod N = x_i^2 - N$ for i 'small' enough.

Let p be a prime, and l an integral exponent, remark that:

$$p^l|(x_i^2 \mod N) \Leftrightarrow x_i^2 - N \equiv O\ [p^l] \Leftrightarrow x_i \equiv \pm r_l\ [p^l] \Leftrightarrow i \equiv -\lfloor\sqrt{N}\rfloor \pm r_l\ [p^l]$$

where r_l denotes a square root of N taken modulo p^l (the other being $-r_l$).

1.1 Quadratic Sieve Method[1]

`Sieve:`

1. Choose a smoothness bound B;
2. Initialise table T to 1;
3.
   - $p = 3$: For all i such that $i \equiv -\lfloor\sqrt{N}\rfloor \pm r_3 \mod 3$, multiply $T[i]$ by 3.
   - $p^l = 9$: For all i such that $i \equiv -\lfloor\sqrt{N}\rfloor \pm r_9 \mod 9$, multiply $T[i]$ by 3 ($r_9 = 1; -r_9 = 8$).
   - ...
4. Repeat for all (or at least a bunch) of p^l, $p \leq B$;
5. At the end of the process, the x_i such that $T[i] \approx 2i\sqrt{N}$ are expected to factor over B whereas the x_i such that $T[i] << 2$ are expected not to factor. And there are $\approx \frac{2\tau}{p^k}$ congruence classes $mod(p^k)$ between 1 and τ.

Total cost: $\sum_{p\in\mathbb{P}}\sum_k \frac{2\tau}{p^k} \leq \sum_{p\in\mathbb{P}, p\leq B}\sum_k \frac{2\tau}{p^k} + \sum_{p' \text{ prime}}\sum_{k=2}^{log\ N} \frac{2\tau}{p'^k} = 2\tau.log\ log\ B + \mathcal{O}(\tau.log\ N)$

Amortised cost: $\mathcal{O}(log\ log\ B + log\ N) = B^{o(1)}$ since $log\ N = \mathcal{O}((log\ B)^2)$

NB: It is inadvisable to implement the algorithm as described above as there will be problems with memory space and with multiplications. One solution is, instead of initialising to 1 and multiplying by p, to initialise to 0 and add an approximation of $log\ p$. You can then correct approximation

[1] The QS algo was used to crack the RSA challenge (*Scientific American 1977*) in 1994 after 7 months and using 1600 computers in parallel.

errors by fine tuning the bound $T[i] \approxeq 2i\sqrt{N}$. This will yield a few (but manageably few) false positives and negatives.

The QS method (as all congruence based algorithms) parallelises well, as we can split T into independant sub-tables. Unfortunately the linear algebra steps cannot be parallelised as well.

There are several optimisations on QS:

- Multiple Polynomials Quadratic Sieve (MPQS)
- Large prime variation (PMPQS)
- Double large prime variation (PPMPQS)

1.2 Large Prime Variation

Whenever we find a *partial relation* of the form $x_i^2 = \left(\prod p_{ij}^{u_i}\right).P$, with $B \leq P < B^2$, we keep it. If we find another partial relation with the same prime P, combine them to obtain one[2] of the following relations:

$$\begin{cases} \left(\frac{x_i}{x_j}\right)^2 = \left(\prod p_{ij}^{u_i - u_j}\right) \mod N \\ \text{OR} \\ \left(\frac{x_i x_j}{p}\right)^2 = \left(\prod p_{ij}^{u_i + u_j}\right) \mod N \end{cases}$$

1.3 Double Large Prime Variation

First, keep track of all *partial-partial relations* of the form $x_i^2 = \left(\prod p_{ij}^{u_i}\right).P_1P_2$, with $B \leq P_1, P_2 < B^2$. Note that this requires 'to recognise the decomposition P_1P_2' (using ECM, ρ-Pollard, P-pollard...).

Then build a multigraph whose vertices are the primes P_i that appear in the partial-partial relations, and where $P_i \sim P_j$ if there is a partial-partial relation with P_iP_j. Use a union-find structure to find the cycles[3] and take the product of the relations corresponding to the edges of the cycle in order to get the relation:

$$\left(\frac{x_{i_1} \dots x_{i_k}}{\prod P_i}\right)^2 = \prod p_{ij}^{u_i}$$

1.3.1 Number Field Sieve

Two variations:

1. General Number Field Sieve (GNFS)[4]

[2] There is no point in taking both as they are the same if the exponents are taken modulo 2.
[3] In practice, we only need to find a 'basis' of the cycles.
[4] In 1999, RSA-155 (512 bits) was factored in six months using GNFS.

2. Special Number Field Sieve (SNFS): specialises in factoring $N = r^e \pm s$, for e, s small.

The key is taking $\alpha := \frac{1}{d}$, where d is a parameter to be tuned. The optimal d is $\mathcal{O}\left(\left(\frac{log\ N}{log\ log\ N}\right)^{1/3}\right)$, which yields a time complexity of $exp(C.(log\ N)^{1/3}.(log\ log\ N)^{2/3})$, where $\begin{cases} C_{SNFS} = \sqrt[3]{32/9} \\ C_{GNFS} = \sqrt[3]{64/9} \end{cases}$

The NFS algorithm is state-of-the-art, and research has mostly moved on to the discrete logarithm problem.

2 Discrete Logarithm

2.1 The Problem

Definition 1 (Discrete Logarithm)**.** *Let G be a finite[5] cyclic group, generated by g.*

Define the discrete logarithm *function as* $DL_g : \begin{array}{ccl} G & \to & \mathbb{Z}/N\mathbb{Z} \\ h & \mapsto & x\ s.t.\ g^x = h \end{array}$ *, where $N := \#G$.*

$\triangleright$ *Note that since $G = <g>$, such an x is uniquely defined for each $h \in G$.*

Definition 2 (Diffie-Hellman (DH))**.** *(g and N are public parameters)*

- Conditional Diffie-Hellman problem (CDH)
  - ***Input:*** g^x, g^y
  - ***Task:*** *Compute* g^{xy}
- Decisional Diffie-Hellman problem (DDH)
  - ***Input:*** g^x, g^y, g^z
  - ***Task:*** *Decide whether* $z \equiv xy \pmod N$

There are abelian groups, described in the early 2000s, in which the DDH problem is solved, but not the CDH problem: pairings, which provided the first good identity-based cryptography. However the security assumptions are still shaky and sometimes contradictory when all stacked together...

Theorem 3. *If we break DLP, we break CDH.*

Remark 4. *There are partial reductions from DLP to CDH:*

- *Non-uniform reductions (Maurer and Wolf), related to the existence of nice elliptic curves: Maurer and Wolf [1, 2] proved that for every group G with prime order p, DLP reduces to CDH if we are able to find an elliptic curve over $\mathbb{F}_p$ with smooth order, and Muzereau, Smart, and Vercauteren [3] showed that such an elliptic curve exists for the various elliptic curve groups recommended by standards;*

[5] All infinite cyclic groups are isomorphic to $\mathbb{Z}$, however there is no *good* way of representing the elements of an infinite cyclic group using a reasonable and finite number of bits. Even the 'smaller' elements could prove problematic.

- *Sub-exponential reductions in some cases.*

Theorem 5. *If we break CDH, we break DDH.*

The DH protocol:

Alice		Bob
Sample $x \in \mathbb{Z}/N\mathbb{Z}$	$\xrightarrow{g^x}$	
	$\xleftarrow{g^y}$	Sample $y \in \mathbb{Z}/N\mathbb{Z}$
Compute $(g^y)^x$		Compute $(g^x)^y$

An eavesdropping attacker Eve has access only to g^x and g^y and therefore must solve CDH to gain access to the secret g^{xy}.

The El Gamal discrete log encryption scheme[6]:

- `Secret key:` $a \in \mathbb{Z}/N\mathbb{Z}$
- `Public key:` g^a
- `Encryption function:` $Enc(m) = (g^r, (g^a)^r\, m),$ with $r \hookleftarrow \mathcal{U}\,(\mathbb{Z}/N\mathbb{Z})$
- `Decryption function:` $Dec(x, y) = y.x^{-a}$

▷ Note that $Dec(Enc(m)) = (g^a)^r m (g^r)^{-a} = m$ as G is cyclic and therefore abelian.

Theorem 6. *If the DDH problem is hard, then the El Gamal encryption scheme is IND-CPA[7] secure.*

Summary:

The DDH assumption is stronger than the DL assumption, and we can classify the groups according to the (presumed) hardness of the DL problem:

- The easy groups: $(\mathbb{Z}/N\mathbb{Z}, +)$
- The easy-ish groups: the Galois Field[8] $(GF(p^n), \times)$ with n large and p a small prime
- The reasonably hard groups: $(GF(p), \times)$ (ie. $(\mathbb{Z}/p\mathbb{Z})^*$) with p a large prime
- The very hard groups: elliptic curves (genus 2 hyperelliptic)

[6] The El Gamal encryption scheme is used in GPG.

[7] To guarantee IND-CPA security, we require in the security game that the challenger only treat *correct* encryption requests (ie. fail if the attacker requests something else than $EncQuery(M_0, M_1)$ with $M_0, M_1 \in G$).

[8] The French notation for $GF(p^n)$ is $\mathbb{F}_{p^n}^*$

2.2 The Attacks

2.2.1 Generic algoritms

1. *Exhaustive method:* try all possible values of x; Complexity $\mathcal{O}(N)$
2. *Pohlig-Hellman method:*

   Theorem 7 (Informal Statement)**.** *Assume that we know the the prime factor decomposition $N = \prod_{i=1}^{r} p_i^{e_i}$. Then DL in a group of order N reduces to e_1 instances of DL in groups (of the same nature) of order p_1* ***and*** *e_2 instances of DL in groups of order p_2 ...*

   Proof. The reducion is comprised of two steps: (a) DL_N reduces to $DL_{p_1^{e_1}}$, $DL_{p_2^{e_2}}$, ..., and $DL_{p_r^{e_r}}$ and (b) $DL_{p_i^{e_i}}$ reduces to e_i instances of DL_{p_i}

   (a) Define $g_i := g^{N/p_i^{e_i}}$ and $h_i := h^{N/p_i^{e_i}}$. Note that if $x \equiv DL_g(h) \pmod{N}$ then $g_i^x = h$, so $DL_{g_i}(h_i) \equiv x \pmod{p_i^{e_i}}$. It follows[9] that $DL_g(h) \equiv DL_{g_i}(h_i) \pmod{p_i^{e_i}}$, so by the chinese remainders theorem $DL_g(h)$ can be recovered from $DL_{g_i}(h_i)$.

   (b) Define $\overline{g} := g^{p^{e-1}}$ and $\overline{h} := h^{p^{e-1}}$. As before, if $x \equiv DL_g(h) \pmod{p^e}$ then $x \equiv DL_{\overline{g}}(\overline{h}) \pmod{p}$. Define $\alpha := DL_{\overline{g}}(\overline{h})$, $h' := h.g^{-\alpha}$, and $g' := g^p$.
   Since there is an X s.t. $\begin{cases} h = g^X \\ X \equiv \alpha \pmod{p} \end{cases}$ we have $h.g^{-\alpha} = g^{X-\alpha} = (g')^{\frac{X-\alpha}{p}} \in < g' >$.
   Therefore, $DL_g(h) = DL_{\overline{g}}(\overline{h}) + p.DL_{g'}(h') \pmod{p^e}$, where $DL_{\overline{g}}(\overline{h})$ is a call to DL in a group of order p and $DL_{g'}(h')$ is a call to DL in a group of order p^{e-1} (so by induction $(e-1)$ calls to DL in groups of order p).

   □

   Bottom-line: DL in G is only as hard as the largest prime factor of $\#G$.
   Caveat: The above is only true as long as you use generic algorithms for a 'generic group' (if there is such a thing ...), but specific methods can be more efficient in the whole group than in the subgroups called recursively by the method, in which case reducing will not help.

2.2.2 Baby-Step/Giant-Step

Let $K \in [\![0; N]\!]$ be a parameter to be tuned later.

1. *Baby-step:*
   - Compute g^i for $i = 0$ to $(K-1)$
   - Store the (i, g^i) in a suitable data structure (e.g. a hashtable)
2. *Giant-step:*
   - Compute g^K then h, $h.g^{-K}$, $h.g^{-2K}$ ...until a collision $h.g^{-jK} = g^i$ is found for some i, j

[9] because $x \equiv DL_g(h) \pmod{N}$ and $p_i^{e_i} | N$ imply $x \equiv DL_g(h) \pmod{p_i^{e_i}}$

- Output $DL \leftarrow i + jK$

A collision will occur after at most $\frac{N}{K}$ steps. Indeed, by Euclidean division, $DL_g(h) = jK + i$ where $0 \leq i < K$ and $j \leq \frac{DL_g(h)}{K} \leq \frac{N}{K}$.

`Space complexity:` $\mathcal{O}(K)$

`Time complexity:` $\begin{cases} \text{Worst case: } K + \frac{N}{K} \\ \text{Average case: } K + \frac{N}{2K} \end{cases}$

- $K_{opt}^{worst} = \sqrt{N}$ which yields a time complexity of $2\sqrt{N}$
- $K_{opt}^{avg} = \sqrt{N}$ which yields a time complexity of $\sqrt{2N}$

References

[1] Ueli M. Maurer. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete logarithms. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 271–281, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[2] Ueli M. Maurer and Stefan Wolf. The diffie–hellman protocol. *Designs, Codes and Cryptography*, 19(2):147–171, Mar 2000.

[3] A. Muzereau, N. P. Smart, and F. Vercauteren. The equivalence between the dhp and dlp for elliptic curves used in practical applications. *LMS Journal of Computation and Mathematics*, 7:5072, 2004.