Solving Discrete Logarithms with Partial Knowledge of the Key

K. Gopalakrishnan
East Carolina University
Greenville, NC, USA

Nicolas Theriault
Universidad de Talca
Talca, Chile

Chui Zhi Yao
University of California
Riverside, CA, USA

INDOCRYPT'07
Indian Institute of Technology, Madras
12th Dec. 2007

Discrete Log Problem (DLP)

- Let G be a cyclic group of prime order p and let g be a generator of G.

Discrete Log Problem (DLP)

- Let G be a cyclic group of prime order p and let g be a generator of G.
- Given $\beta \in G$, the discrete logarithm problem is to determine α such that $g^\alpha = \beta$.

Discrete Log Problem (DLP)

- Let G be a cyclic group of prime order p and let g be a generator of G.
- Given $\beta \in G$, the discrete logarithm problem is to determine α such that $g^\alpha = \beta$.
- The presumed computational difficulty of solving the DLP in appropriate groups is the basis of many cryptosystems and protocols.

Discrete Log Problem (DLP)

- Let G be a cyclic group of prime order p and let g be a generator of G.
- Given $\beta \in G$, the discrete logarithm problem is to determine α such that $g^{\alpha} = \beta$.
- The presumed computational difficulty of solving the DLP in appropriate groups is the basis of many cryptosystems and protocols.
- The primary reason for the popularity of ECC over RSA is that there are currently no known subexponential algorithms to solve the DLP in groups used in ECC.

Generic Groups

- “Generic Group” refers to a group whose structure we do not know.

Generic Groups

- “Generic Group” refers to a group whose structure we do not know.
- It is presented in the form of a “Black Box”.

Generic Groups

- “Generic Group” refers to a group whose structure we do not know.
- It is presented in the form of a “Black Box”.
- Each group element has a unique encoding.

Generic Groups

- “Generic Group” refers to a group whose structure we do not know.
- It is presented in the form of a “Black Box”.
- Each group element has a unique encoding.
- Given the encoding of two elements g and h of the group, the black box can

Generic Groups

- “Generic Group” refers to a group whose structure we do not know.
- It is presented in the form of a “Black Box”.
- Each group element has a unique encoding.
- Given the encoding of two elements g and h of the group, the black box can
  - produce $g \cdot h$, in unit time.

Generic Groups

- “Generic Group” refers to a group whose structure we do not know.
- It is presented in the form of a “Black Box”.
- Each group element has a unique encoding.
- Given the encoding of two elements g and h of the group, the black box can
  - produce $g \cdot h$, in unit time.
  - decide whether $g = h$, in unit time.

Generic Groups

- "Generic Group" refers to a group whose structure we do not know.
- It is presented in the form of a "Black Box".
- Each group element has a unique encoding.
- Given the encoding of two elements g and h of the group, the black box can
  - produce $g \cdot h$, in unit time.
  - decide whether $g = h$, in unit time.
  - compute any given power of g (including g^{-1}), in time $O(\log p)$.

Algorithms for DLP in Generic Groups

Baby-step Giant-step	Rho	Kangaroo*
Shanks	Pollard	Pollard
Deterministic	Probabilistic	Probabilistic
Time $O(\sqrt{p})$	$O(\sqrt{p})$	$O(\sqrt{b-a})$
Space $O(\sqrt{p})$	$O(1)$	$O(1)$

*Assumes that DL is known to lie in the interval $[a, b]$

Lower Bounds

- Victor Shoup has established a lower bound of $\Omega(\sqrt{p})$ for any probabilistic algorithm that can solve the DLP for generic groups.

Lower Bounds

- Victor Shoup has established a lower bound of $\Omega(\sqrt{p})$ for any probabilistic algorithm that can solve the DLP for generic groups.
- Hence the Baby-step Giant-step method and the rho method are optimal algorithms to solve the DLP and cannot be improved further (except possibly by a constant factor).

Lower Bounds

- Victor Shoup has established a lower bound of $\Omega(\sqrt{p})$ for any probabilistic algorithm that can solve the DLP for generic groups.
- Hence the Baby-step Giant-step method and the rho method are optimal algorithms to solve the DLP and cannot be improved further (except possibly by a constant factor).
- The reason for the popularity of the ECC is that the only known algorithms to solve the DLP over elliptic curve groups are the generic algorithms.

Side Channel Attacks

- "Side Channel Attacks" focus on the implementation of an algorithm rather than the specification to break a cryptosystem.

Side Channel Attacks

- “Side Channel Attacks” focus on the implementation of an algorithm rather than the specification to break a cryptosystem.
- By observing the implementation being executed, the attacker can make correlations between the events that occur in the processor and the data being processed.

Side Channel Attacks

- "Side Channel Attacks" focus on the implementation of an algorithm rather than the specification to break a cryptosystem.
- By observing the implementation being executed, the attacker can make correlations between the events that occur in the processor and the data being processed.
- Famous examples include Timing based attacks and Power Analysis based attacks.

Problem Considered

- We assume that some partial information about the secret key is revealed by side channel attacks.

Problem Considered

- We assume that some partial information about the secret key is revealed by side channel attacks.
- We can certainly ignore the extra information and use a generic algorithm of complexity $O(\sqrt{p})$ to break the system.

Problem Considered

- We assume that some partial information about the secret key is revealed by side channel attacks.
- We can certainly ignore the extra information and use a generic algorithm of complexity $O(\sqrt{p})$ to break the system.
- We can do an exhaustive search using the extra information to break the system.

Problem Considered

- We assume that some partial information about the secret key is revealed by side channel attacks.
- We can certainly ignore the extra information and use a generic algorithm of complexity $O(\sqrt{p})$ to break the system.
- We can do an exhaustive search using the extra information to break the system.
- Can we do something in between? In other words, can we use the partial information intelligently to break the system?

Nature of Partial Information

- We consider two different scenarios based on the nature of partial information revealed.

Nature of Partial Information

- We consider two different scenarios based on the nature of partial information revealed.
- In the first scenario, we assume that a sequence of contiguous bits of the key is revealed.

Nature of Partial Information

- We consider two different scenarios based on the nature of partial information revealed.
- In the first scenario, we assume that a sequence of contiguous bits of the key is revealed.
- In the second scenario, we assume that incomplete information about the "Square and Multiply Chain" (used to efficiently exponentiate) is revealed.

Contiguous bits are known

The goal is to come up with an algorithm whose complexity will be square root of the size of the remaining key space (and thus optimal)

Contiguous bits are known

The goal is to come up with an algorithm whose complexity will be square root of the size of the remaining key space (and thus optimal)

Left Part is Known Can simply use Kangaroo Algorithm.

Contiguous bits are known

The goal is to come up with an algorithm whose complexity will be square root of the size of the remaining key space (and thus optimal)

Left Part is Known Can simply use Kangaroo Algorithm.

Right Part is Known Can be solved in optimal time as shown by Teske.

Contiguous bits are known

The goal is to come up with an algorithm whose complexity will be square root of the size of the remaining key space (and thus optimal)

Left Part is Known Can simply use Kangaroo Algorithm.

Right Part is Known Can be solved in optimal time as shown by Teske.

Middle Part is Known Has not been studied in the Literature.

Middle Part is Known

- Assume that we know M and N such that

$$\alpha = \alpha_1 MN + \alpha_2 M + \alpha_3$$

where $0 \leq \alpha_2 < N$ is known and with $0 \leq \alpha_3 < M$.

Middle Part is Known

- Assume that we know M and N such that

$$\alpha = \alpha_1 MN + \alpha_2 M + \alpha_3$$

  where $0 \le \alpha_2 < N$ is known and with $0 \le \alpha_3 < M$.

- Suppose, we are given an integer r, $0 < r < p$, such that we can write rMN as $kp + s$ with $|s| < p/2$. Then,

$$\begin{aligned} r\alpha &= r\alpha_1 MN + r\alpha_2 M + r\alpha_3 \\ &= \alpha_1 kp + s\alpha_1 + r\alpha_2 M + r\alpha_3 \\ &= \alpha_1 kp + r\alpha_2 M + \alpha' \end{aligned}$$

Reducing to Kangaroo Algorithm

$$
\begin{aligned}
g^{\alpha r} &= g^{\alpha_1 kp + r\alpha_2 M + \alpha'} \\
(g^{\alpha})^r &= (g^p)^{\alpha_1 k} g^{r\alpha_2 M} g^{\alpha'} \\
\beta^r &= g^{r\alpha_2 M} g^{\alpha'}
\end{aligned}
$$

Reducing to Kangaroo Algorithm

$$\begin{aligned} g^{\alpha r} &= g^{\alpha_1 kp + r\alpha_2 M + \alpha'} \\ (g^{\alpha})^r &= (g^p)^{\alpha_1 k} g^{r\alpha_2 M} g^{\alpha'} \\ \beta^r &= g^{r\alpha_2 M} g^{\alpha'} \end{aligned}$$

- Denoting $\left(\beta \times g^{-\alpha_2 M}\right)^r$ by β', the above equation can be written in the form $\beta' = g^{\alpha'}$.

Reducing to Kangaroo Algorithm

$$
\begin{aligned}
g^{\alpha r} &= g^{\alpha_1 kp + r\alpha_2 M + \alpha'} \\
(g^{\alpha})^r &= (g^p)^{\alpha_1 k} g^{r\alpha_2 M} g^{\alpha'} \\
\beta^r &= g^{r\alpha_2 M} g^{\alpha'}
\end{aligned}
$$

- Denoting $\left(\beta \times g^{-\alpha_2 M}\right)^r$ by β', the above equation can be written in the form $\beta' = g^{\alpha'}$.
- Note that β' can be computed from β as r, α_2, and M are known.

Reducing to Kangaroo Algorithm

$$\begin{aligned} g^{\alpha r} &= g^{\alpha_1 kp + r\alpha_2 M + \alpha'} \\ (g^{\alpha})^r &= (g^p)^{\alpha_1 k} g^{r\alpha_2 M} g^{\alpha'} \\ \beta^r &= g^{r\alpha_2 M} g^{\alpha'} \end{aligned}$$

- Denoting $\left(\beta \times g^{-\alpha_2 M}\right)^r$ by β', the above equation can be written in the form $\beta' = g^{\alpha'}$.
- Note that β' can be computed from β as r, α_2, and M are known.
- Invoke Kangaroo Algorithm to compute $\alpha^{'}$.

Bounding Interval Size

- When s is positive, $\alpha' = \alpha_3 r + \alpha_1 s$ must be in the interval

$$\left[0, r(M-1) + s\left(\frac{p}{MN} - 1\right)\right]$$

Bounding Interval Size

- When s is positive, $\alpha' = \alpha_3 r + \alpha_1 s$ must be in the interval

$$\left[0, r(M-1) + s\left(\frac{p}{MN} - 1\right)\right]$$

- Similary, if s is negative, α' must be in the interval

$$\left[s\left(\frac{p}{MN} - 1\right), r(M-1)\right]$$

Bounding Interval Size

- When s is positive, $\alpha' = \alpha_3 r + \alpha_1 s$ must be in the interval

$$\left[0, r(M-1) + s\left(\frac{p}{MN} - 1\right)\right]$$

- Similary, if s is negative, α' must be in the interval

$$\left[s\left(\frac{p}{MN} - 1\right), r(M-1)\right]$$

- In both cases we can restrict the value of α' to an interval of length $rM + |s|\frac{p}{MN}$.

Minimizing Interval Size

- We want to select the parameters r and s such that the interval size is minimized.

Minimizing Interval Size

- We want to select the parameters r and s such that the interval size is minimized.
- We can do so, by using Dirichlet's Theorem on rational approximations.

Minimizing Interval Size

- We want to select the parameters r and s such that the interval size is minimized.
- We can do so, by using Dirichlet's Theorem on rational approximations.
- We can guarantee that the size of the interval is at most $O(2p/\sqrt{N})$.

Minimizing Interval Size

- We want to select the parameters r and s such that the interval size is minimized.
- We can do so, by using Dirichlet's Theorem on rational approximations.
- We can guarantee that the size of the interval is at most $O(2p/\sqrt{N})$.
- So, the time complexity of the overall algorithm will be $O(\sqrt{2}p^{1/2}/N^{1/4})$.

Remarks

- Once α' is known, we can solve the easy diophantine equation $\alpha' = r\alpha_3 + s\alpha_1$ and extract α_1 and α_3.

Remarks

- Once α' is known, we can solve the easy diophantine equation $\alpha' = r\alpha_3 + s\alpha_1$ and extract α_1 and α_3.
- Together with the known middle part α_2, we get the discrete log α.

Remarks

- Once α' is known, we can solve the easy diophantine equation $\alpha' = r\alpha_3 + s\alpha_1$ and extract α_1 and α_3.
- Together with the known middle part α_2, we get the discrete log α.
- The complexity that we are able to get is not quite optimal in the general case.

Remarks

- Once α' is known, we can solve the easy diophantine equation $\alpha' = r\alpha_3 + s\alpha_1$ and extract α_1 and α_3.
- Together with the known middle part α_2, we get the discrete log α.
- The complexity that we are able to get is not quite optimal in the general case.
- However, if p is Mersenne prime (or if p is sufficiently close to 2^l), we are able to get optimal complexity.

Square and Multiply Algorithm

For example, to compute g^{43}

- Write 43 in binary as 101011.

Square and Multiply Algorithm

For example, to compute g^{43}

- Write 43 in binary as 101011.
- Replace each 0 by S (Square) and 1 by SM (Square and Multiply) to get SMSSMSSMSM.

Square and Multiply Algorithm

For example, to compute g^{43}

- Write 43 in binary as 101011.
- Replace each 0 by S (Square) and 1 by SM (Square and Multiply) to get SMSSMSSMSM.
- Start with $h = 1$ and do the operations (Squaring h and Multiplying h by g) specified by the above string from left to right, storing the result back in h each time.

Square and Multiply Algorithm

For example, to compute g^{43}

- Write 43 in binary as 101011.
- Replace each 0 by S (Square) and 1 by SM (Square and Multiply) to get SMSSMSSMSM.
- Start with $h = 1$ and do the operations (Squaring h and Multiplying h by g) specified by the above string from left to right, storing the result back in h each time.
- $1 \longrightarrow 1 \longrightarrow g \longrightarrow g^2 \longrightarrow g^4 \longrightarrow g^5 \longrightarrow g^{10} \longrightarrow g^{20} \longrightarrow g^{21} \longrightarrow g^{42} \longrightarrow g^{43}$

Assumptions

- In the second scenario, we assume that (incomplete) information about the "Square and Multiply Chain" is revealed by the side channel attacks. Specifically, we assume that

Assumptions

- In the second scenario, we assume that (incomplete) information about the “Square and Multiply Chain” is revealed by the side channel attacks. Specifically, we assume that
  - the total length n of the square and multiply chain is known.

Assumptions

- In the second scenario, we assume that (incomplete) information about the “Square and Multiply Chain” is revealed by the side channel attacks. Specifically, we assume that
  - the total length n of the square and multiply chain is known.
  - the number m of M’s is known.

Assumptions

- In the second scenario, we assume that (incomplete) information about the “Square and Multiply Chain” is revealed by the side channel attacks. Specifically, we assume that
  - the total length n of the square and multiply chain is known.
  - the number m of M’s is known.
  - Exact positions of $m - i$ of the M’s are known.

Assumptions

- In the second scenario, we assume that (incomplete) information about the “Square and Multiply Chain” is revealed by the side channel attacks. Specifically, we assume that
  - the total length n of the square and multiply chain is known.
  - the number m of M’s is known.
  - Exact positions of $m - i$ of the M’s are known.
- The problem is to utilize the partial information and figure out the entire chain.

Observations

- We need to figure out the exact positions of the remaining i M's. Then, the entire chain is determined.

Observations

- We need to figure out the exact positions of the remaining i M's. Then, the entire chain is determined.
- We can do an exhaustive search in time $O(n^i)$. Guess the positions of the i M's and then check whether the guess is right.

Observations

- We need to figure out the exact positions of the remaining i M's. Then, the entire chain is determined.
- We can do an exhaustive search in time $O(n^i)$. Guess the positions of the i M's and then check whether the guess is right.
- We can make use of the fact that every M should be preceded and followed by a S. But, this will not affect the asymptotics.

A solution using an additional assumption

- Suppose that somehow we can split the chain into two parts such that $i/2$ M's are on the left part and the remaining $i/2$ M's are on the right part.

$$\begin{aligned}\beta &= g^{\alpha} = g^{a2^x+b} \\ &= \left(g^{2^x}\right)^a g^b\end{aligned}$$

A solution using an additional assumption

- Suppose that somehow we can split the chain into two parts such that $i/2$ M's are on the left part and the remaining $i/2$ M's are on the right part.

$$\begin{aligned} \beta &= g^{\alpha} = g^{a2^x+b} \\ &= \left(g^{2^x}\right)^a g^b \end{aligned}$$

- If we denote g^{-2^x} by h, then the above equation reduces to

$$\beta^{-1} \times g^b = h^a$$

Algorithm

- Make a guess a for the left part.

Algorithm

- Make a guess a for the left part.
- Compute h^a.

Algorithm

- Make a guess a for the left part.
- Compute h^a.
- Record the pair (a, h^a) in a table.

Algorithm

- Make a guess a for the left part.
- Compute h^a.
- Record the pair (a, h^a) in a table.
- Repeat the above for each possible guess a.

Algorithm

- Make a guess a for the left part.
- Compute h^a.
- Record the pair (a, h^a) in a table.
- Repeat the above for each possible guess a.
- Sort the table based on second column.

Algorithm

- Make a guess a for the left part.
- Compute h^a.
- Record the pair (a, h^a) in a table.
- Repeat the above for each possible guess a.
- Sort the table based on second column.
- Space Complexity is $O(n^{i/2})$ ignoring logarithmic terms.

Algorithm - Contd.

- Make a guess b for the right part.

Algorithm - Contd.

- Make a guess b for the right part.
- Compute $y = \beta^{-1} \times g^b$.

Algorithm - Contd.

- Make a guess b for the right part.
- Compute $y = \beta^{-1} \times g^b$.
- Check if y is in the second column of some row of the table.

Algorithm - Contd.

- Make a guess b for the right part.
- Compute $y = \beta^{-1} \times g^b$.
- Check if y is in the second column of some row of the table.
- If so, the guess is correct, and the corresponding a is in the first column.

Algorithm - Contd.

- Make a guess b for the right part.
- Compute $y = \beta^{-1} \times g^b$.
- Check if y is in the second column of some row of the table.
- If so, the guess is correct, and the corresponding a is in the first column.
- If not, the guess is wrong and we make another guess for b until we succeed.

Algorithm - Contd.

- Make a guess b for the right part.
- Compute $y = \beta^{-1} \times g^b$.
- Check if y is in the second column of some row of the table.
- If so, the guess is correct, and the corresponding a is in the first column.
- If not, the guess is wrong and we make another guess for b until we succeed.
- Time Complexity is $O(n^{i/2})$ ignoring logarithmic terms.

Getting rid of the assumption

- We don't really need the additional assumption.

Getting rid of the assumption

- We don't really need the additional assumption.
- There are only $O(n)$ ways of dividing the chain into two parts.

Getting rid of the assumption

- We don't really need the additional assumption.
- There are only $O(n)$ ways of dividing the chain into two parts.
- One of them must have the property that $i/2$ M's are in each part. So, we try each way of splitting the chain, one position at a time.

Getting rid of the assumption

- We don't really need the additional assumption.
- There are only $O(n)$ ways of dividing the chain into two parts.
- One of them must have the property that $i/2$ M's are in each part. So, we try each way of splitting the chain, one position at a time.
- The overall complexity of the algorithm will be $O(n^{1+\lfloor \frac{i}{2} \rfloor})$.

Conclusion and Open Problems

- Under two different scenarios of partial information we have better algorithms to find the discrete log.

Conclusion and Open Problems

- Under two different scenarios of partial information we have better algorithms to find the discrete log.
- At present, we don't know how to solve the discrete log problem efficiently when the bits revealed are not in contiguous postions.

Conclusion and Open Problems

- Under two different scenarios of partial information we have better algorithms to find the discrete log.
- At present, we don't know how to solve the discrete log problem efficiently when the bits revealed are not in contiguous postions.
- Sometimes, one uses the NAF (Non-adjacent Form) representation to do the exponentiation. If we know partial information about the NAF we don't know how to solve the discrete log problem efficiently.