Схема Шнорра с укороченной ЭЦП и предварительным хэшированием *

С.В. Агиевич

1 Схема

Схемой электронной цифровой подписи (ЭЦП) в современной криптографии принято называть набор из трех алгоритмов: `Gen`, `Sign` и `Verify`.

Алгоритм `Gen` является вероятностным. Он берет на вход параметр безопасности l и выдает долговременные параметры $params$, личный ключ d и соответствующий ему открытый ключ Q. Параметр l определяет размерности долговременных параметров и ключей. Чем больше l, тем выше уровень стойкости алгоритмов ЭЦП.

Алгоритм `Sign` также является вероятностным. Алгоритм берет на вход сообщение X, долговременные параметры $params$, личный ключ d и выдает электронную цифровую подпись s.

Алгоритм `Verify` берет на вход сообщение X, долговременные параметры $params$, открытый ключ Q, подпись s и выдает результат `ДА` или `НЕТ`. Ответ `ДА` означает, что подпись к сообщению X была выработана с использованием личного ключа d и сообщение X не было изменено с момента выработки подписи. Ответ `НЕТ` означает противное.

Схема ЭЦП является корректной, если равенство

$$\texttt{Verify}(X, params, Q, \texttt{Sign}(X, params, d)) = \texttt{ДА}$$

выполняется для любых допустимых сообщений X и троек $(params, d, Q)$, полученных с помощью алгоритма `Gen`.

Схема Шнорра [5] является одной из наиболее эффективных и теоретически обоснованных схем ЭЦП. На ее основе построен стандарт Республики Беларусь СТБ 1176.2-99, южнокорейские стандарты KCDSA и EC-KCDSA.

В работе предлагается следующая редакция схемы Шнорра:

1. Алгоритм `Gen` по заданному уровню стойкости l строит аддитивную группу и находит ее элемент G, который имеет порядок q, $2^{2l-1} < q < 2^{2l}$. Элемент G порождает циклическую группу $\langle G\rangle = \{0, G, 2G, \dots, (q-1)G\}$.

   Алгоритм `Gen` определяет также функции хэширования $h\colon \mathcal{X} \to \{0, 1, \dots, 2^{2l} - 1\}$ и $\varphi\colon \{0, 1, \dots, 2^{2l} - 1\} \times \langle G\rangle \to \{2^l, 2^l + 1, \dots, 2^{l+1} - 1\}$. Здесь $\mathcal{X}$ — множество допустимых сообщений X.

   Личный ключ d выбирается наудачу из множества $\{1, 2, \dots, q-1\}$. Открытый ключ определяется как $Q = dG$.

2. Алгоритм `Sign` состоит из следующих шагов:

   1) выбрать наудачу k из множества $\{1, 2, \dots, q-1\}$;

*Версия 2013.04.12

2) $R \leftarrow kG$;

3) $s_0 \leftarrow \varphi(h(X), R)$;

4) $s_1 \leftarrow (k - s_0 d - h(X)) \bmod q$;

5) возвратить (s_0, s_1).

По окончании алгоритма $s_0 Q + (s_1 + h(X))G = (s_0 d + k - s_0 d - h(X) + h(X))G = R$ и, следовательно,

$$\varphi(h(X), s_0 Q + (s_1 + h(X))G) = s_0. \tag{1}$$

3. В алгоритме `Verify` подпись $s = (s_0, s_1)$ к сообщению X признается действительной, если выполняется равенство (1).

Обсудим особенности предлагаемой схемы.

Длина ЭЦП. Первая часть подписи (s_0) кодируется l-битовым словом, вторая часть (s_1) — $2l$-битовым словом и в целом длина ЭЦП составляет $3l$ битов. Для сравнения, в стандартах СТБ 1776.2-99, ГОСТ Р 34.10-1994, ГОСТ Р 34.10-2012, DSA, ECDSA, KCDSA, EC-KCDSA на паритетных уровнях стойкости вырабатывается $4l$-битовая подпись. Отметим, что самую короткую на сегодняшний день подпись — $2l$ битов — дает схема, предложенная в работе [2]. К сожалению, схема не получила широкого распространения, поскольку при выработке и проверке ЭЦП приходится выполнять трудоемкие вычисления, связанные со спариваниями на эллиптических кривых.

Уменьшение длины ЭЦП связано с «обрезкой» s_0 с $2l$ до l битов относительно СТБ 1176.2-99, KCDSA и других известных реализаций схемы Шнорра. Фактически «обрезка» была введена в [5], но в дальнейшем по непонятным причинам от нее отказались. Отметим, что в недавней работе [3] «обрезка» была рекомендована как способ повышения эффективности схемы Шнорра при сохранении криптографической стойкости.

«Обрезка» s_0 не только сокращает длину ЭЦП, но и ускоряет вычисление $s_0 Q + (s_1 + h(X))G$ при проверке подписи. Например, при использовании бинарных методов нахождения кратных элементов группы $\langle G \rangle$ проверка ЭЦП будет выполняться примерно на 8% медленнее выработки. Для сравнения, без «обрезки» проверка замедляется на 17%.

Защита от повтора подписи. При «обрезке» время построения коллизии для φ уменьшается до $O(2^{l/2})$ операций. В связи с этим в выражение для определения s_1 и в проверочное соотношение (1) введены слагаемые $\pm h(X)$. Если бы этих слагаемых не было, то владелец личного ключа смог бы построить одинаковые подписи для двух различных сообщений X и X' за время порядка $2^{l/2}$. Впоследствии владелец мог бы отказаться от факта подписания X', утверждая, что подписывал X. Для проведения атаки владелец выбирает различные X и X' и случайное k до тех пор, пока не будет построена коллизия $\varphi(h(X), kG) = \varphi(h(X'), kG)$. При этом числа $s_0 = \varphi(X, kG)$ и $s_1 = (k - s_0 d) \bmod q$ образуют действительную ЭЦП как для X, так и для X'.

Предварительное хэширование. В алгоритмах `Sign` и `Verify` сообщение X предварительно хэшируется с помощью функции h. Предварительное хэширование упрощает встраивание алгоритмов ЭЦП в современные информационные системы. Дело в том, что программные интерфейсы таких систем построены по принципу hash-and-sign. Это значит, что на вход программных функций, реализующих алгоритмы ЭЦП, подаются не сами сообщения X, а их хэш-значения $h(X)$. Например, функция выработки ЭЦП программного интерфейса Microsoft CSP так и называется: `CPSignHash` («подписать хэш-значение»).

2 Стойкость

Схема ЭЦП считается криптографически надежной, если для нее вычислительно трудно решить задачу «единичная подделка». Противник, который решает данную задачу, получает в свое распоряжение гипотетическое устройство (назовем его π-*ящиком*), реализующее алгоритмы `Gen`, `Sign`, `Verify`. До передачи π-ящика противнику выполнен алгоритм `Gen` и выходные данные алгоритма использованы для инициализации устройства: в защищенной памяти π-ящика сохранен личный ключ d, а в открытой памяти сохранены параметры $params$ и открытый ключ Q. Противник может читать открытую память, т. е. противник знает $params$ и Q.

Алгоритмы `Sign`, `Verify` доступны противнику через *оракулов* $\mathcal{S}$ и $\mathcal{V}$. В криптографической литературе под оракулом понимается вспомогательный алгоритм, который выполняется за один шаг работы основного алгоритма. Другими словами, при оценке трудоемкости атак противника время работы оракула не учитывается, учитывается только число обращений к нему. Входные данные, которые передаются оракулу, называются вопросом, выходные — ответом.

Оракул $\mathcal{S}$ получает вопрос X, читает из памяти $params$ и d и возвращает ответ $\texttt{Sign}(X, params, d)$. Оракул $\mathcal{V}$ получает вопрос (X, s_0, s_1), читает из памяти $params$ и Q и возвращает ответ $\texttt{Verify}(X, params, Q, (s_0, s_1))$.

Противник может задавать оракулам любые вопросы, получать и анализировать ответы. Задача «единичная подделка» состоит в построении вопроса (X, s_0, s_1) такого, что

1) $\mathcal{V}(X, s_0, s_1) = \texttt{ДА}$;

2) вопрос X до этого не задавался оракулу $\mathcal{S}$.

Будем считать, что противник — это некоторый вероятностный алгоритм A. Работа алгоритма заканчивается формированием искомого вопроса (X, s_0, s_1), обращением к $\mathcal{V}$ с этим вопросом и возвратом соответствующего ответа. Пусть $\mathbf{Adv}(A)$ — вероятность того, что будет получен ответ `ДА`.

При обосновании стойкости требуется показать, что для любого противника A с разумными ограничениями на его ресурсы, вероятность $\mathbf{Adv}(A)$ мала. Для этого доказывают, что алгоритм A, который решает задачу «единичная подделка», может быть преобразован (с приемлемыми издержками) в алгоритм, который решает другую задачу T. В таких случаях говорят, что задача T сводится к задаче «единичная подделка». Сводимость означает, что если T — трудная задача, то «единичная подделка» также трудна.

В качестве T используем составную задачу $\mathbf{DL}(G) \vee \mathbf{Coll}(h)$ (решить $\mathbf{DL}(G)$ или $\mathbf{Coll}(h)$), где

1) $\mathbf{DL}(G)$ — задача дискретного логарифмирования в $\langle G \rangle$: по G и $Q \in \langle G \rangle$ определить число $d \in \{0, 1, \ldots, q-1\}$ такое, что $R = dG$;

2) $\mathbf{Coll}(h)$ — задача построения коллизии для h: найти различные $X, X' \in \mathcal{X}$ такие, что $h(X) = h(X')$.

Если $\langle G \rangle$ и h выбраны как криптографически надежные группа и хэш-функция соответственно, то для решения каждой из задач требуется время порядка 2^l.

Следующая теорема определяет сведе́ние $\mathbf{DL}(G) \vee \mathbf{Coll}(h)$ к задаче «единичная подделка» в модели ROM (Random Oracle Model). В этой модели π-ящик снабжается еще одним оракулом $\mathcal{H}$, который реализует вычисление значений φ. К оракулу $\mathcal{H}$ обращаются оракулы $\mathcal{S}$ и $\mathcal{V}$ когда им необходимо получить значение φ. Модель ROM идеализирует функцию φ: оракул $\mathcal{H}$ выбирает ответ на каждый новый вопрос случайно равновероятно независимо от других ответов.

При доказательстве теоремы использован подход, сформированный в [4] и развитый в [1]. Похожая теорема сформулирована в [3] для классической схемы Шнорра (без предварительного хэширования).

Теорема. Пусть алгоритм A с вероятностью $\mathbf{Adv}(A)$ решает задачу «единичная подделка» в модели ROM, используя β_H обращений к оракулу $\mathcal{H}$ и β_S обращений к оракулу $\mathcal{S}$. Пусть

$$\varepsilon = \frac{\beta_H\beta_S + \beta_S(\beta_S - 1)/2}{q} < 1.$$

Тогда существует алгоритм B, который с вероятностью

$$\mathbf{Adv}(B) \geq \mathbf{Adv}(A)\left(\frac{\mathbf{Adv}(A)}{\beta_H} - \frac{1}{2^l}\right)(1-\varepsilon)^2$$

решает $\mathbf{DL}(G) \vee \mathbf{Coll}(h)$, используя 2 обращения к A и еще $O(\beta_H + \beta_S)$ операций. При этом

$$\mathbf{Adv}(A) \leq \frac{\beta_H}{2^l} + \frac{\sqrt{\beta_H\,\mathbf{Adv}(B)}}{1-\varepsilon}.$$

Список литературы

[1] Bellare M., Neven G. Multi-signatures in the plain public-key model and a general forking lemma // In: ACM CCS 06, ACM Press, 2006, 390–399.

[2] Boneh D., Lynn B., Shacham H. Short Signatures from the Weil Pairings // Advances in Cryptology — ASIACRYPT'01 Proceedings, LNCS 2248, 514-532, 2001.

[3] Neven G., Smart N. P., Warinschi B. Hash Function Requirements for Schnorr Signatures // Journal of Mathematical Cryptology, vol. 3, no. 1, 2009, 69–87.

[4] Pointcheval D., Stern J. Security arguments for digital signatures and blind signatures // Journal of Cryptology, vol. 13, no. 3, 2000, 361–396.

[5] Schnorr C. P. Efficient Signature Generation by Smart Cards // J. Cryptology, vol. 4, no. 3, 1991, 161–174.

Приложение. Доказательство теоремы

Опишем функционирование π-ящика, который реализует предложенную схему ЭЦП (запись $u \xleftarrow{R} U$ означает случайный равновероятный выбор u из множества U):

<table>
<tr><td colspan="3">π-ящик с оракулом $\mathcal{H}$</td></tr>
<tr>
<td>Оракул $\mathcal{H}$ (хэширование (H,R)):

1. Возвратить $\varphi(H,R)$.</td>
<td>Оракул $\mathcal{S}$ (выработка ЭЦП для X):

1. $H \leftarrow h(X)$.
2. $k \xleftarrow{R} \{0,1,\ldots,q-1\}$.
3. $R \leftarrow kG$.
4. $s_0 \leftarrow \mathcal{H}(H,R)$.
5. $s_1 \leftarrow (k - ds_0 - H) \bmod q$.
6. Возвратить (s_0,s_1).</td>
<td>Оракул $\mathcal{V}$ (проверка ЭЦП (s_0,s_1) для X):

1. $H \leftarrow h(X)$.
2. $R \leftarrow s_0Q + (s_1+H)G$.
3. Если $\mathcal{H}(H,R) \neq s_0$, то возвратить <code>НЕТ</code>.
4. Возвратить <code>ДА</code>.</td>
</tr>
</table>

Модель ROM состоит в изменении способа функционирования оракула $\mathcal{H}$:

π-ящик в модели ROM		
Оракул $\mathcal{H}$:	Оракул $\mathcal{S}$:	Оракул $\mathcal{V}$:
1. Если вопрос (H, R) уже задавался, то возвратить предыдущий ответ. 2. $s_0 \xleftarrow{R} \{2^l, \ldots, 2^{l+1} - 1\}$. 3. Возвратить s_0.	1. $H \leftarrow h(X)$. 2. $k \xleftarrow{R} \{0, 1, \ldots, q-1\}$. 3. $R \leftarrow kG$. 4. $s_0 \leftarrow \mathcal{H}(H, R)$. 5. $s_1 \leftarrow (k - ds_0 - H) \bmod q$. 6. Возвратить (s_0, s_1).	1. $H \leftarrow h(X)$. 2. $R \leftarrow s_0 Q + (s_1 + H)G$. 3. Если $\mathcal{H}(H, R) \neq s_0$, то возвратить `НЕТ`. 4. Возвратить `ДА`.

Оракул $\mathcal{H}$ реализует случайный равновероятный независимый выбор образов функции φ. В начале взаимодействия с π-ящиком образы не определены, по мере взаимодействия функция φ понемногу «проявляется».

Рассмотрим следующий *поддельный* π-ящик, который симулирует работу предыдущего (*настоящего*):

поддельный π-ящик		
Оракул $\mathcal{H}$:	Оракул $\mathcal{S}$:	Оракул $\mathcal{V}$:
1. Если вопрос (H, R) уже задавался, то возвратить предыдущий ответ. 2. $s_0 \xleftarrow{R} \{2^l, \ldots, 2^{l+1} - 1\}$. 3. Возвратить s_0.	1. $H \leftarrow h(X)$. 2. $s_0 \xleftarrow{R} \{2^l, \ldots, 2^{l+1} - 1\}$. 3. $s_1 \xleftarrow{R} \{0, 1, \ldots, q-1\}$. 4. $R \leftarrow s_0 Q + (s_1 + H)G$. 5. Если вопрос (H, R) уже задавался $\mathcal{H}$ и был получен ответ $\neq s_0$, то `СТОП`. 6. Зафиксировать ответ s_0 оракула $\mathcal{H}$ на вопрос (H, R). 7. Возвратить (s_0, s_1).	1. $H \leftarrow h(X)$. 2. $R \leftarrow s_0 Q + (s_1 + H)G$. 3. Если $\mathcal{H}(H, R) \neq s_0$, то возвратить `НЕТ`. 4. Возвратить `ДА`.

В поддельном π-ящике изменен способ функционирования оракула $\mathcal{S}$ — оракул не использует личный ключ d и не задает $\mathcal{H}$ вопросы, а сам формирует на них ответы. Оракул $\mathcal{S}$ поддельного ящика может выполнять дополнительную команду `СТОП`, которая приводит к остановке вычислений. После выполнения команды поведение π-ящика не определено (оно нас и не будет интересовать).

Несмотря на изменения, статистически ответы оракулов поддельного и настоящего π-ящиков неотличимы вплоть до вызова команды `СТОП`. Вызов команды инициируют следующие события:

1. Оракул $\mathcal{H}$ дал ответ s_0 на вопрос (H, R), оракул $\mathcal{S}$ формирует ответ (s'_0, s'_1) на вопрос X',

при этом оказывается, что $H = h(X')$, $R = s'_0 Q + (s'_1 + H)G$, но $s_0 \neq s'_0$. Вероятность этого события

$$\leq \mathsf{P}\left\{R = s'_0 Q + (s'_1 + H)G\right\} \mathsf{P}\left\{s_0 \neq s'_0\right\} = \frac{1}{q}\left(1 - \frac{1}{2^l}\right) < \frac{1}{q}.$$

2. Оракул $\mathcal{S}$ формирует ответы (s_0, s_1) и (s'_0, s'_1) на вопросы X и X' (возможно одинаковые), при этом оказывается, что $h(X) = h(X')$, $s_0 Q + (s_1 + h(X))G = s'_0 Q + (s'_1 + h(X'))G$, но $s_0 \neq s'_0$. Вероятность этого события снова меньше $1/q$.

Если оракулам $\mathcal{H}$ и $\mathcal{S}$ поддельного π-ящика задано β_H и β_S вопросов соответственно, то вероятность появления хотя бы одного из перечисленных событий строго меньше величины

$$\frac{\beta_H \beta_S + \beta_S(\beta_S - 1)/2}{q}.$$

Понятно, что при $\beta_S, \beta_H \ll \sqrt{q}$ добиться выполнения команды `СТОП` можно лишь с весьма малой вероятностью.

Вернемся к заявленному в формулировке теоремы сведéнию. Пусть алгоритму B (будем называть его *симулятором*) требуется либо решить уравнение $dG = Q$ относительно d, либо найти коллизию для h. Для этого симулятор использует алгоритм A, который решает задачу «единичная подделка», взаимодействуя с оракулами π-ящика.

Будем считать, что A задает в точности $\beta_H \geq 1$ вопросов оракулу $\mathcal{H}$ и в точности β_S вопросов оракулу $\mathcal{S}$ (если алгоритму потребовалось меньше вопросов, то он может задать несколько дополнительных, игнорируя ответы на них). Будем снабжать вопросы и ответы оракула $\mathcal{H}$ индексами $[i]$: $(H[i], R[i])$ — i-й вопрос, $s_0[i]$ — i-й ответ. Будем считать, что перед выдачей решения A проверяет его, обращаясь к оракулу $\mathcal{V}$. Алгоритм A является вероятностным и использует в своей работе случайную ленту ω, которой управляет симулятор.

Для работы A нужен π-ящик, снаряженный ключами d и Q. Но симулятор не знает d, как раз этот ключ требуется найти. Поэтому B передает A поддельный π-ящик, предполагая, что за время его использования A команда `СТОП` не будет вызвана.

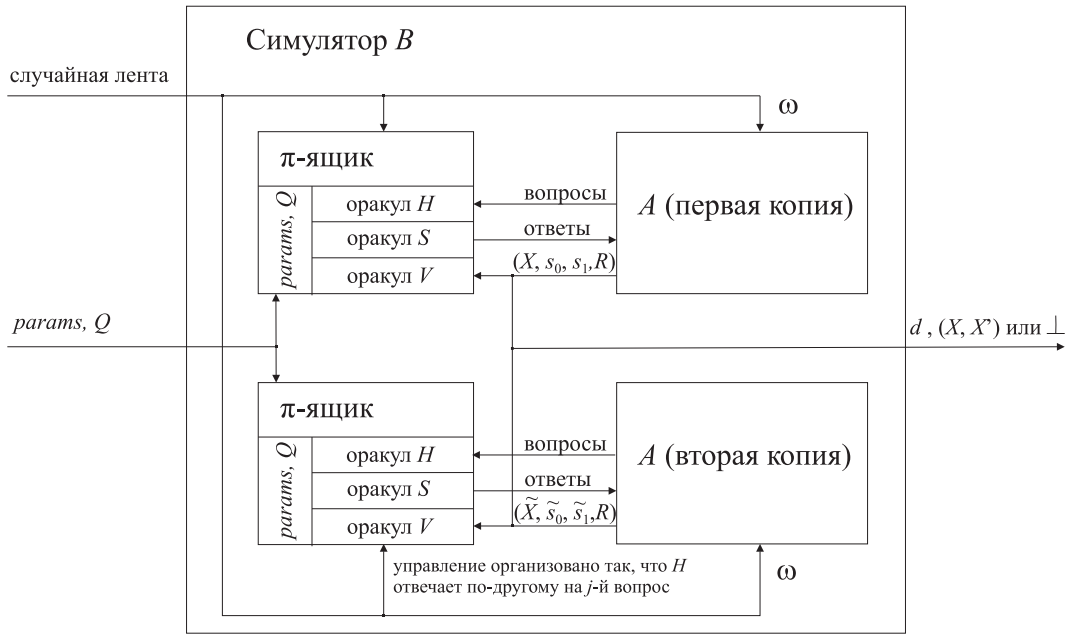

Рис. 1: Симулятор B

Симулятор B действует следующим образом (см. рисунок 1):

1. Запускает первую копию алгоритма A со случайной лентой ω. Копии передается первый π-ящик, ответы оракулов которого формируются по случайной ленте B. Если алгоритм завершился по команде `СТОП` или не нашел решение задачи «единичная подделка», то B прекращает работу с отрицательным результатом $\perp$.

2. Пусть (X, s_0, s_1, R) — решение, полученное первой копией A. Ответ $\mathcal{V}(X, s_0, s_1, R) =$ `ДА` означает равенство $\mathcal{H}(H, R) = s_0$, где $H = h(X)$. Это возможно в следующих ситуациях:

   1) вопрос (H, R) до этого не задавался $\mathcal{H}$, т. е. подпись была просто угадана;
   2) вопрос (H, R) до этого задавался $\mathcal{H}$ и был получен ответ s_0;
   3) ответ s_0 на вопрос (H, R) был зафиксирован оракулом $\mathcal{S}$ при выработке ЭЦП к сообщению X'. По условию задачи «единичная подделка» $X \neq X'$, т. е. построена коллизия $h(X) = H = h(X')$ для h.

   Если выполняется третий случай, то симулятор возвращает решение (X, X') задачи $\mathbf{Coll}(h)$.

3. Пусть $(h(X), R) = (H[j], R[j])$, т. е. найденное первой копией решение построено по j-му вопросу оракулу $\mathcal{H}$. Используя свою случайную ленту, симулятор создает второй π-ящик, j-эквивалентный первому. Имеется в виду, что оракулы $\mathcal{H}$ этих π-ящиков дают одинаковые ответы вплоть до вопроса $(H[j], R[j])$. На этот вопрос оракул первого ящика дает ответ $s_0[j]$, а оракул второго — ответ $\tilde{s}_0[j] \neq s_0[j]$.

4. Запускает вторую копию алгоритма A со случайной лентой ω. Копии передается второй π-ящик. Если алгоритм завершился по команде `СТОП` или не нашел решение задачи «единичная подделка», то B прекращает работу с отрицательным результатом $\perp$.

5. Пусть $(\tilde{X}, \tilde{s}_0, \tilde{s}_1, \tilde{R})$ — решение, полученное второй копией. Симулятор проверяет решение также, как и решение первой копии. Если вопрос $(h(\tilde{X}), \tilde{R})$ не задавался оракулу $\mathcal{H}$, то B находит решение $(\tilde{X}, \tilde{X}')$ задачи $\mathbf{Coll}(h)$ и возвращает его.

6. Пусть $(\tilde{h}(X), \tilde{R}) = (\tilde{H}[\tilde{j}], \tilde{R}[\tilde{j}])$, т. е. найденное второй копией решение построено по $\tilde{j}$-му вопросу оракулу $\mathcal{H}$ второго π-ящика. Если $\tilde{j} \neq j$, то симулятор прекращает работу с отрицательным результатом $\perp$.

7. Обе копии алгоритма были заправлены одинаковой случайной лентой и до j-го вопроса оракулу $\mathcal{H}$ выполнялись совершенно аналогично. Поэтому $(H[j], R[j]) = (\tilde{H}[j], \tilde{R}[j])$. Но как раз на j-м вопросе оракулы двух π-ящиков дали различные ответы $s_0[j]$ и $\tilde{s}_0[j]$. Это значит, что
$$s_0[j]Q + (s_1 + H[j])G = R[j] = \tilde{s}_0[j]Q + (\tilde{s}_1 + \tilde{H}[j])G.$$
   Отсюда
$$s_0[j]d + s_1 \equiv \tilde{s}_0[j]d + \tilde{s}_1 \pmod{q}$$
   и B определяет искомый дискретный логарифм
$$d = (s_1 - \tilde{s}_1)(\tilde{s}_0[j] - s_0[j])^{-1} \bmod q.$$

Подведем итоги и укажем связь между вероятностями успеха алгоритмов A и B. Введем в рассмотрение следующие события:

$$\mathcal{E}_0 = \{\text{одна из копий } A \text{ завершена по команде } \texttt{СТОП}\},$$
$$\mathcal{E}_1 = \{\text{алгоритм } B \text{ нашел решение } \mathbf{Coll}(h)\},$$
$$\mathcal{E}_2 = \{\text{алгоритм } B \text{ нашел решение } \mathbf{DL}(G)\}.$$

Тогда

$$\begin{aligned}\mathbf{Adv}(B) &= \mathsf{P}\left\{\mathcal{E}_1\bar{\mathcal{E}}_0\right\} + \mathsf{P}\left\{\mathcal{E}_2 \mid \bar{\mathcal{E}}_1\bar{\mathcal{E}}_0\right\}\mathsf{P}\left\{\bar{\mathcal{E}}_1\bar{\mathcal{E}}_0\right\} \geq \\ &\geq \mathsf{P}\left\{\mathcal{E}_2 \mid \bar{\mathcal{E}}_1\bar{\mathcal{E}}_0\right\}\mathsf{P}\left\{\bar{\mathcal{E}}_0\right\} \geq \\ &\geq \mathsf{P}\left\{\mathcal{E}_2 \mid \bar{\mathcal{E}}_1\bar{\mathcal{E}}_0\right\}(1-\varepsilon)^2.\end{aligned}$$

Для оценки вероятности $\alpha = \mathsf{P}\left\{\mathcal{E}_2 \mid \bar{\mathcal{E}}_1\bar{\mathcal{E}}_0\right\}$ воспользуемся обобщенной леммой о разветвлении (general forking lemma) из [1]. Получим

$$\alpha \geq \mathbf{Adv}(A)\left(\frac{\mathbf{Adv}(A)}{\beta_H} - \frac{1}{2^l}\right),$$

откуда следует первая оценка.

Вторая оценка получается в результате арифметических преобразований первой.