Individuelle Schlüsselverifikation via Socialist Millionaires' Protocol

Sven Moritz Hallberg
sm@khjk.org

Juni 2008

Zusammenfassung

Das zentrale Problem jeder Public-Key-Infrastruktur besteht darin, für alle Teilnehmer die Echtheit der von ihnen bezogenen öffentlichen Schlüssel sicherzustellen. Kann das nicht gewährleistet werden, wird typischerweise das gesamte System durch Man-in-the-Middle-Angriffe verwundbar. Übliche Herangehensweisen sind der Einsatz zentraler Zertifizierungsstellen (SSL), verteilte Zertifizierungsmodelle (PGP-Web-of-Trust) oder der direkte Abgleich von Schlüsselabdrücken (Fingerprints). Ein von Boudot et al. entwickeltes Zero-Knowledge-Protokoll ermöglicht den geheimen und sicheren Vergleich zweier Abdrücke über einen ungesicherten Kanal. Dabei benötigen die Kommunikationspartner nur einmalig ein vergleichsweise schwaches Shared-Secret.

Einführung

Folgende Aufgabe ist bekannt als *Yaos Millionärsproblem*[10]: Zwei Millionäre wünschen zu bestimmen, wer der reichere ist, ohne dabei ihre Kontostände offenzulegen. Eine Abwandlung davon ist das sogenannte *Socialist Millionaires' Problem*[5], bei dem die beiden lediglich daran interessiert sind, ob ihr Reichtum *gleich* ist.

Formal ist in beiden Fällen eine Möglichkeit gefragt, wie Alice und Bob einen Funktionswert $f(x, y, \ldots)$ bestimmen können, ohne daß einer von ihnen zusätzliche Informationen über die Eingabewerte $(x, y, \ldots)$ erhält. Ein Verfahren mit dieser Eigenschaft wird als *Zero-Knowledge-Protokoll* bezeichnet. Ist das Ergebnis von f ein Wahrheitswert, so handelt es sich um einen *Zero-Knowledge-Beweis*. Weiterhin nennt man ein Protokoll *fair*, wenn keiner der Teilnehmer es mit nennenswertem Informationsvorsprung beenden kann.

Es existieren Lösungen für das Millionärsproblem[5, 7, 9], aus denen man trivial auch eine Lösung für die "sozialistische" Variante erhält. Allerdings wächst die Komplexität all dieser Verfahren mit der Größe der Eingabewerte. Die jüngste Lösung[9] von Shundong et al. etwa benötigt noch $\mathcal{O}(max(x, y))$ Operationen.

Die von Boudot et al. entwickelte Lösung des Socialist Millionaires' Problem[3] kommt mit $\mathcal{O}(1)$ modularen Exponentiationen aus und eignet sich damit für folgendes Szenario: Alice und Bob wollen verschlüsselt kommunizieren und haben einander über einen ungesicherten Kanal ihre öffentlichen Schlüssel P_A und P_B gesendet. Dabei empfängt Alice $P_{\tilde{B}}$ und Bob $P_{\tilde{A}}$. Um sicherzugehen, daß $P_{\tilde{A}} = P_A$ und $P_{\tilde{B}} = P_B$ sind, vereinbaren sie ein Kennwort s und führen den Zero-Knowledge-Beweis, daß

$$h(P_A, P_{\tilde{B}}, s) = h(P_{\tilde{A}}, P_B, s)$$

gilt. Dabei sei h eine kryptographische Streufunktion. Durch das Kennwort s ist sichergestellt, dass nur Alice und Bob das Protokoll durchführen können. Dank der Zero-Knowledge-Eigenschaft des Protokolls wird nichts über s bekannt und ein Angreifer muß das Kennwort zur Laufzeit des Protokolls erraten. Dadurch kann ein relativ "schwaches" Wort gewählt werden. Unter geeigneten Umständen können Alice und Bob sich mittels Umschreibungen direkt über den ungesicherten Kanal auf ein hinreichend sicheres s verständigen.

Ein Verfahren nach obigem Prinzip wurde von Borisov et al. unter der Bezeichnung "Socialist Millionaires' Protocol" (SMP) für das Programm Off-the-Record implementiert[1, 2]. Off-the-Record ermöglicht die verschlüsselte Kommunikation über Instant-Messaging-Dienste wie ICQ. Dabei bietet sich das hier vorgestellte Verfahren insbesondere an, weil die Kommunikationspartner häufig miteinander vertraut sind aber in der Regel über keinen bestehenden Kanal zur sicheren Schlüsselverifizierung verfügen.

Voraussetzungen

Das vorgestellte Protokoll ist ein Diffie-Hellman-artiges Verfahren. Das heißt, es basiert auf der Schwierigkeit, in einer gegebenen endlichen Gruppe Logarithmen zu berechnen, genauer auf dem Diffie-Hellman- bzw. Diffie-Hellman-Entscheidungsproblem. Beide sind nicht schwerer zu lösen als ein diskreter Logarithmus. Daher müssen alle drei Probleme für die Sicherheit des Protokolls als *schwer* vorausgesetzt werden.

Im folgenden sei G_q eine endliche Gruppe der Ordnung q, q sei eine große Primzahl. Durch g sei ein Erzeuger von G_q gegeben. Auftretende Exponenten $(a, b, c, \ldots)$ sind aus $\mathbb{Z}$ (bzw. $\mathbb{Z}/q\mathbb{Z}$).

Definition (DL)**.** Die Aufgabe, aus $g^a \in G_q$ auf $a \in \mathbb{Z}$ zu schließen, heißt *Problem des diskreten Logarithmus.*

Definition (DH)**.** Die Aufgabe, aus beliebigen $g^a, g^b \in G_q$ auf g^{ab} zu schließen, heißt *Diffie-Hellman-Problem*

Bemerkung. DH ist äquivalent dazu, aus g^a und g^{ab} auf g^b zu schließen.

Definition (DDH)**.** Die Aufgabe, für beliebige $g^a, g^b, g^c \in G_q$ zu entscheiden, ob $g^c = g^{ab}$ gilt, heißt *Diffie-Hellman-Entscheidungsproblem.*

Bemerkung. DDH ist äquivalent dazu, für g^a, g^c und g^{ab} zu entscheiden, ob $g^c = g^b$.

Zero-knowledge Bausteine

Das Socialist Millionaires' Protocol greift auf einige bekannte Zero-Knowledge Beweisverfahren zurück, auf die hier nur verwiesen werden soll. Es handelt sich in allen Fällen um *nicht-interaktive* Verfahren.

Kenntnis eines diskreten Logarithmus Dieses Verfahren geht auf Schnorr[8] zurück. Alice präsentiert Bob Werte y, g und wünscht, ihm zu beweisen, daß sie ein $x \in \mathbb{Z}/q\mathbb{Z}$ kennt, so daß $y = g^x$. Dazu wählt sie ein zufälliges $r \in \mathbb{Z}/q\mathbb{Z}$ und schickt Bob die Werte $c := h(g^r)$ und $D := r - xc$ als Beweis. Bob ist überzeugt, falls $c = h(g^D y^c)$.

Kenntnis diskreter Koordinaten Eine einfache Erweiterung des vorigen Protokolls stammt von Okamoto[6]. Alice nennt g_1, g_2 und behauptet, x_1, x_2 zu kennen, s.d. $y = g_1^{x_1} g_2^{x_2}$. Sie wählt zufällige r_1, r_2 und sendet als Beweis $c =$

$h(g_1^{r_1}, g_2^{r_2})$ sowie $D_1 = r_1 - x_1 c$ und $D_2 = r_2 - x_2 c$. Bob ist überzeugt, falls $c = h(g_1^{D_1} g_2^{D_2} y^c)$.

Gleichheit zweier diskreter Logarithmen Ein Verfahren von Chaum und Pedersen[4] läßt Alice zu gegebenen $g_1, g_2, y \in G_q$ demonstrieren, daß $log_{g_1} y = log_{g_2} y$. Sie wählt ein zufälliges $r \in \mathbb{Z}/q\mathbb{Z}$ und schickt Bob als Beweis $c := h(g_1^r, g_2^r)$ und $D := r - xc$. Bob ist überzeugt, falls $c = h(g_1^D y^c, g_2^D, y^c)$.

Motivation

G_q sei eine Gruppe wie oben, g_1 sei ein Erzeuger. Die zu vergleichenden Eingaben von Alice und Bob werden mit x bzw. y bezeichnet. Sie können als Elemente von $\mathbb{Z}/q\mathbb{Z}$ vorausgesetzt werden. Ansonsten verwende man ihre Werte unter einer geeigneten Streufunktion.

Die Grundidee des Verfahrens besteht darin, den direkten Vergleich

$$x = y \quad \Leftrightarrow \quad x - y = 0$$

mittels einer injektiven Funktion f zu verschleiern:

$$f(x - y) = f(0) \quad \Rightarrow \quad x - y = 0$$

Als erster Ansatz für f bietet sich die Exponentiation in der Gruppe G_q an:

$$\frac{g_1^x}{g_1^y} = g_1^{x-y} = g_1^0 = 1$$

Das Ergebnis des Vergleiches könnte also von Alice und Bob nach Austausch von g_1^x und g_1^y berechnet werden. Aus diesen Werten kann ein Dritter zwar x und y nicht mehr direkt berechnen, aber es läßt sich noch für beliebige Kandidaten entscheiden, ob sie zu den ausgetauschten Werten passen. Die Zero-Knowledge-Eigenschaft wäre somit nicht gegeben.

Um das "Durchprobieren" von x bzw. y zunächst für passive Angreifer auszuschließen, wird g_1 durch einen Erzeuger g_3 ersetzt, der nur Alice und Bob bekannt ist. Hierzu führen sie einen gewöhnlichen Diffie-Hellman-Schlüsselaustausch durch: Alice sendet $g_1^{x_a}$ (x_a zufällig), Bob antwortet mit $g_1^{x_b}$ (x_b zufällig) und beide berechnen $(g_1^{x_a})^{x_b} = (g_1^{x_b})^{x_a} =: g_3$. Das System erhielte damit die Form

$$\frac{g_3^x}{g_3^y} = g_3^{x-y} = g_3^0 = 1 \quad .$$

Gegen einen aktiven Angriff seitens Alice oder Bob[1] folgt ein zunächst unzureichender Ansatz, der sich allerdings zum Ziel führen lässt. Alice wählt eine weitere Zufallszahl a, mit der sie x "maskiert" (Bob verfährt analog): Ausgetauscht werden nun g_3^{a+x} und g_3^{b+y}. Wir erhalten die Gleichung

$$\frac{g_3^{a+x}}{g_3^{b+y}} = g_3^{(a+x)-(b+y)} = g_3^{a-b+x-y} = g_3^{a-b} = \frac{g_3^a}{g_3^b} \quad .$$

Wie man an der rechten Seite erkennt, müssen nun zusätzlich g_3^a und g_3^b ausgetauscht werden. Damit könnte Bob jedoch wieder auf $g_3^x = g_3^{a+x}/g_3^a$ schließen.

Zur Lösung dieses Problems, schreiben wir die Gleichung zunächst als

$$\frac{g_3^{a+x}}{g_3^{b+y}} = \left(\frac{g_1^{a+x}}{g_1^{b+y}}\right)^{x_a x_b} = \frac{g_3^a}{g_3^b}$$

[1] Das schließt auch man-in-the-middle-Attacken ein.

und bemerken, daß wir die Exponentiation herauszögern können: Zunächst werden g_1^{a+x} und g_1^{b+y} ausgetauscht, aus denen mithilfe von g_3^a und g_3^b nichts zu gewinnen ist. Danach bilden beide Seiten den Quotienten und potenzieren mit $x_a x_b$ durch einen *zweiten* Diffie-Hellman-Austausch.

Es bleibt noch eine letzte Lücke zu schließen. Schreiben wir nochmals um:

$$\left(\frac{g_1^a \cdot g_1^x}{g_1^b \cdot g_1^y}\right)^{x_a x_b} = \frac{g_3^a}{g_3^b}$$

Dividiert Bob linke durch rechte Seite, so erhält er g_3^{x-y} und Multiplikation mit g_3^y liefert wieder g_3^x. Dieses Möglichkeit läßt sich jedoch leicht unterbinden, indem im Ausdruck g_1^x / g_1^y die Basis g_1 durch einen anderen Erzeuger g_2 ersetzt wird[2]. Wir erhalten damit in Formelform das Verfahren, das sich als sicher erweisen wird:

$$\left(\frac{g_1^a \cdot g_2^x}{g_1^b \cdot g_2^y}\right)^{x_a x_b} = \frac{g_3^a}{g_3^b} \quad , \quad g_3 = g_1^{x_a x_b}$$

Protokollablauf

Alice und Bob einigen sich auf zwei Erzeuger g_1 und g_2 von G_q, wobei $log_{g_1} g_2$ unbekannt sei. Alice wählt $a, x_a \in \mathbb{Z}/q\mathbb{Z}$ zufällig; Bob wählt analog b, x_b. Das Protokoll verläuft dann in drei Schritten:

1. Diffie-Hellman-Austausch zur Erzeugung von g_3
2. Austausch von Zähler und Nenner der zwei Quotienten:
   a) g_3^a und g_3^b (bezeichnet als P_a und P_b in Abb. 1)
   b) $g_1^a g_2^x$ und $g_1^b g_2^y$ (bezeichnet als Q_a und Q_b in Abb. 1)
3. Diffie-Hellman-Austausch zur Potenzierung der linken Seite

Jeder Schritt wird begleitet von Zero-Knowledge-Teilbeweisen (s.o.), die Alice und Bob zusichern, daß die ausgetauschten Werte tatsächlich das Ergebnis der geforderten Berechnungen darstellen. Abbildung 1 zeigt ein detailiertes Schema.

Fairness

Eine leichte Erweiterung läßt das oben beschriebene Protokoll fair werden, so daß keine der beiden Parteien es frühzeitig abbrechen kann, um das Ergebnis für sich zu behalten. Da diese Eigenschaft im Szenario von OTR nicht entscheidend ist, sei die nötige Erweiterung hier nur kurz umrissen. Für Details siehe [3].

Alice und Bob einigen sich auf einen dritten Erzeuger g_0 von G_q, so daß für alle $0 < i \neq j < 3$ die Logarithmen $log_{g_i} g_j$ unbekannt sind. Im ersten Schritt des Protokolls wählen beide eine zusätzliche Zufallszahl e bzw. f von k Bit Länge. Bei der berechnung von P_a bzw. P_b fügen sie einen zusätzlichen Faktor an.

$$\begin{aligned} P_a &:= g_3^a g_0^e \\ P_b &:= g_3^b g_0^f \end{aligned}$$

Der Rest des Protokollablaufs bleibt unverändert bis zur abschließenden Verifikation. Dort haften diese Faktoren der Rechten Seite (P_a/P_b) des Ergebnisses an. Bevor der Vergleich stattfinden kann, müssen sie auch auf die linke Seite gebracht werden. Alice und Bob geben einander nun abwechselnd je *ein einzelnes Bit* von e bzw. f preis, womit sie schließlich den Vergleich durchführen können. Bricht einer von beiden den Austausch ab, hat er einen Informationsvorsprung von höchstens einem Bit.

[2] Es leuchtet ein, daß dessen Logarithmus zur Basis g_1 nicht bekannt sein darf – ein solches Element können Alice und Bob jedoch konstruieren.

Alice				Bob
$x_a \in \mathbb{Z}/q\mathbb{Z}$ zufällig				$x_b \in \mathbb{Z}/q\mathbb{Z}$ zufällig
$g_a := g_1^{x_a} \neq^! 1$				$g_b := g_1^{x_b} \neq^! 1$
	$\leftarrow$	g_a, g_b	$\rightarrow$	
		Schnorr		
	$\leftarrow$	Kenntnis von x_a bzw. x_b	$\rightarrow$	
$g_3 := g_b^{x_a} = g_1^{x_a x_b}$				$g_3 := g_a^{x_b} = g_1^{x_a x_b}$
$a \in \mathbb{Z}/q\mathbb{Z}$ zufällig				$b \in \mathbb{Z}/q\mathbb{Z}$ zufällig
$P_a := g_3^a = g_1^{a x_a x_b}$				$P_b := g_3^b = g_1^{b x_a x_b}$
$Q_a := g_1^a g_2^x$				$Q_b := g_1^b g_2^y$
	$\leftarrow$	P_a, Q_a, P_b, Q_b	$\rightarrow$	
		Okamoto		
	$\leftarrow$	Kenntnis von a, x bzw. b, y	$\rightarrow$	
$P_a/P_b = g_1^{(a-b)x_a x_b}$				$P_a/P_b = g_1^{(a-b)x_a x_b}$
$Q_a/Q_b = g_1^{(a-b)} g_2^{x-y}$				$Q_a/Q_b = g_1^{(a-b)} g_2^{x-y}$
$R_a = (Q_a/Q_b)^{x_a}$				$R_b = (Q_a/Q_b)^{x_b}$
	$\leftarrow$	R_a, R_b	$\rightarrow$	
		Chaum-Pedersen		
	$\leftarrow$	Bew. der Gleichheit $log_{g_1} g_a = log_{Q_a/Q_b} R_a$ bzw. $log_{g_1} g_b = log_{Q_a/Q_b} R_b$	$\rightarrow$	
$R_{ab} := R_b^{x_a}$				$R_{ab} := R_a^{x_b}$
$R_{ab} =^? P_a/P_b$		VERIFIKATION		$R_{ab} =^? P_a/P_b$

Abbildung 1: Schematischer Ablauf des Socialist Millionaires' Protocol

Sicherheit & Korrektheit

Das SMP ist korrekt in dem Sinne, daß Alice und Bob am Ende von der Korrektheit des Ergebnisses überzeugt sind. Dies ergibt sich aus den im Protokoll ausgetauschten Beweisen, die beiden Parteien garantieren, daß alle Werte tatsächlich auf die behauptete Art erzeugt wurden. Damit liefert der Vergleich am Ende das gewünschte Ergebnis.

Sicherheit gegen passive Angreifer

Im folgenden nehmen wir ohne Einschränkung der Allgemeinheit Bob als passiven Angreifer an. Alle Argumente gelten symmetrisch auch für Alice als unehrliche Partei.

Um zu zeigen, daß Bob keinerlei Informationen über x erhalten kann, führen wir dies zum Widerspruch zu den Voraussetzungen an DH bzw. DDH.

Dazu zunächst eine Abstraktion. Im Verlaufe des Protokolls erfährt Bob insgesamt folgende Werte von Alice:

$$\begin{aligned} g_a &= g_1^{x_a} \\ P_a &= g_3^a \\ Q_a &= g_1^a g_2^x \\ R_a &= (Q_a/Q_b)^{x_a} \end{aligned}$$

Da ihm x_b, b und y bekannt sind, lassen sich diese reduzieren zu:

$$\begin{aligned} g_1^{x_a} & \\ g_1^{a x_a} &= P_a^{x_b^{-1}} \\ g_1^a g_2^{x-y} &= g_1^a g_2^x / g_2^y \\ g_2^{(x-y)x_a} &= R_a \cdot P_b^{x_b^{-1}} / g_1^{a x_a} \end{aligned}$$

Setzt man dafür kurz $g_1 = g$, $g_2^{x-y} = g^w$, $x_a = u$ und $a = v$, so ergibt sich, daß Bob die Werte g^u, g^{uv}, g^{uw} und g^{v+w} erhält.

Satz. *Ist DH schwer, so ist es für Bob auch schwer, aus g^u, g^{uv}, g^{uw} und g^{v+w} Alices Geheimnis x zu berechnen.*

Beweis. Wegen $g^w = g_2^{x-y} = g_2^x / g_2^y$ reicht es zu zeigen, daß die Berechnung von g^w schwer ist. Bemerke: $v = a$ und $u = x_a$ sind zufällig gewählt. Außerdem gilt $w = log_{g_1} g_2^{x-y}$, wobei $log_{g_1} g_2$ als unbekannt vorausgesetzt war. Daher müßte Bob zur Berechnung von g^w ein Verfahren für *beliebige* $u, v, w \in$ kennen. Daß ein solches ebenfalls schwer sein muß, besagt das folgende Lemma.

Lemma. *Ist DH schwer, so auch die Berechnung von g^w aus g^u, g^{uv}, g^{uw} und g^{v+w} für beliebige $u, v, w \in \mathbb{Z}/q\mathbb{Z}$.*

Beweis. Angenommen, wir haben ein "Orakel" zur Lösung der gegebenen Aufgabe. Seien g^a, g^{ab} gegeben und $c \in \mathbb{Z}/q\mathbb{Z}$ beliebig gewählt. Mit $a = u$, $b = v$, $c - b = w$ gelten $g^{uw} = g^{ac}/g^{ab}$ und $g^{v+w} = g^c$. Das Orakel liefert uns damit $g^{c-b} = g^c/g^b$, woraus wir g^b erhalten. Damit löst das hypothetische Orakel DH. DH ist also nicht schwerer als die gestellte Aufgabe.

Satz. *Ist DDH schwer, so ist es für Bob auch schwer, zu entscheiden, ob ein gegebener Testwert $\tilde{x}$ gleich x ist.*

Beweis. Setze $g^t = g_2^{\tilde{x}-y}$. Analog zu oben reicht es zu zeigen, daß es schwer ist, zu entscheiden, ob $g^t = g^w$. D.h. Bob bräuchte ein Verfahren, um für beliebige $t, u, v, w \in \mathbb{Z}/q\mathbb{Z}$ aus Kenntnis von g^t, g^u, g^{uv}, g^{uw} und g^{v+w} zu entscheiden, ob $g^t = g^w$. Daß ein solches schwer sein muß, besagt das folgende Lemma.

Lemma. *Ist DDH schwer, so auch die Entscheidung, ob $g^t = g^w$ aus gegebenen Werten g^t, g^u, g^{uv}, g^{uw} und g^{v+w}, für beliebige $t, u, v, w \in \mathbb{Z}/q\mathbb{Z}$.*

Beweis. Angenommen wir haben ein Orakel zur Lösung der gegebenen Aufgabe. Seien g^a, g^{ab}, g^c gegeben und $d \in \mathbb{Z}/q\mathbb{Z}$ beliebig gewählt. Mit $a = u$, $b = v$, $d - c = t$, $d - b = w$ gelten $g^{uw} = g^{ad}/g^{ab}$ und $g^{v+w} = g^d$. Das Orakel nennt uns, ob $g^t = g^w \Leftrightarrow d - c = d - b \Leftrightarrow c = b$. Damit löst das hypothetische Orakel DDH. DDH ist also nicht schwerer als die gestellte Aufgabe.

Sicherheit gegen aktive Angreifer

Zur Sicherheit des Protokolls gegen aktive Angriffe soll hier nur das Beweisprinzip vorgestellt werden.

Angenommen, Bob wäre in der Lage, aus dem Ablauf des Protokolls etwas über x zu erfahren. Dabei darf er sich als aktiver Angreifer beliebig verhalten.

Es wird gezeigt, daß Bob (als Turingmaschine betrachtet) benutzt werden kann, um DH zu lösen.

Für den ausgeführten Beweis siehe [3].

Literatur

[1] *Off-the-Record Messaging Protocol version 2*. `http://www.cypherpunks.ca/otr/Protocol-v2-3.1.0.html`.

[2] Borisov, Nikita, Ian Goldberg und Eric Brewer: *Off-the-record communication, or, why not to use PGP*. In: *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, Seiten 77–84, New York, NY, USA, 2004. ACM.

[3] Boudot, Fabrice, Berry Schoenmakers und Jacques Traoré: *A fair and efficient solution to the socialist millionaires' problem*. Discrete Applied Mathematics, 111(1–2):23–36, 2001.

[4] Chaum, David und Torben Pryds Pedersen: *Wallet databases with observers*. Lecture Notes in Computer Science, 740:89–105, 1993.

[5] Jakobsson, Markus und Moti Yung: *Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers*. Lecture Notes in Computer Science, 1109:186–200, 1996.

[6] Okamoto, Tatsuaki: *Provably secure and practical identification schemes and corresponding signature schemes*. Lecture Notes in Computer Science, 740:31–53, 1993.

[7] Schneier, Bruce: *Applied Cryptography*. Wiley & Sons, 1996.

[8] Schnorr, Claus Peter: *Efficient signature generation by smart cards*. Journal of Cryptology, 4(3):161–174, 1991.

[9] Shundong, Li, Wang Daoshun, Dai Yiqi und Luo Ping: *Symmetric cryptographic solution to Yao's millionaires' problem and an evaluation of secure multiparty computations*. Inf. Sci., 178(1):244–255, 2008.

[10] Yao, Andrew: *Protocols for secure computations*. In: *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, Seiten 160–164, 1982.