Datenschutzfreundliche Terminplanung

Benjamin.Kellermann@tu-dresden.de

3. Dezember 2009

Terminplanungstools sind vielen vielleicht aus Groupwaresystemen bekannt. Ein in letzter Zeit sehr beliebtes Beispiel, welches Terminplanung als stand-alone Web 2.0-Anwendung umsetzt, ist doodle. Doodle sowie andere Lösungen haben das Problem, dass die Vorlieben bzw. Verfügbarkeitsmuster der an der Terminplanung beteiligten Personen veröffentlicht werden. In diesem Beitrag werden ein Protokoll und eine Anwendung vorgestellt, welche das Problem mit homomorpher Verschlüsselung, eine auch im E-Voting verwendete Technik, vermeiden.

1 Einleitung

Web 2.0-Anwendungen haben in den letzten Jahren viel Aufmerksamkeit bekommen. Eine sehr spezielle Web 2.0-Anwendung ist doodle [1]. Mit dieser Anwendung kann man Termine in kleineren Gruppen, wie z. B. einen Kinoabend oder eine Telefonkonferenz planen. Wie bei den meisten Web 2.0-Anwendungen ist auch hier Datenschutz kein Primärziel. Nimmt man an einer Terminabstimmung teil, so teilt man dem Server personenbezogene Daten über seine eigene Verfügbarkeit an verschiedenen Zeitpunkten mit. Aus diesen *Verfügbarkeitsmustern* kann man auf zwei Arten Informationen bekommen. Zum einen können Dritte direkt Daten über das private Leben der Personen einsehen („Wird mein Mann für das Datum an unserem Hochzeitstag stimmen?“). Zum anderen können Dritte diese Informationen mit anderen Informationsquellen verketten und dadurch Individuen identifizieren, die sonst anonym geblieben wären („Das Verfübarkeitsmuster des Benutzers *häschen23* sieht dem meines Arbeitskollegen aber verdächtig ähnlich!“)

In diesem Beitrag wird ein Protokoll vorgestellt, welches eine datenschutzfreundliche Terminplanung realisieren kann. Im 2. Abschnitt werden Grundlagen erklärt, die zum Verstehen des Protokolls benötigt werden. Auf das eigentliche Protokoll wird in Abschnitt 3 eingegangen.

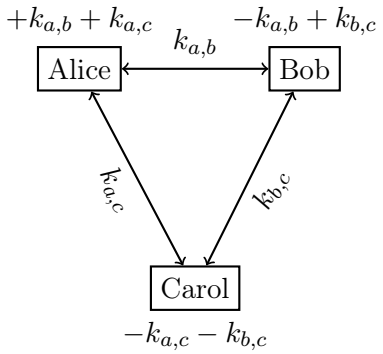


Abbildung 1: Schlüsselaustausch im DC-Netz

2 Grundlagen

2.1 DC-Netz

Das DC-Netz, auch als überlagerndes Senden bekannt, ist ein Protokoll, welches anonyme Kommunikation ermöglicht. 1988 wurde von David Chaum das „dining cryptographers”-Problem vorgestellt in dem Kryptographen anonym ermitteln wollen, wer die Restaurantrechung bezahlt hat [2]. Im gleichen Beitrag wird die ein Protokoll vorgestellt, welches dieses Problem löst. Im Folgenden soll dieses Protokoll vereinfacht anhand eines Beispiels erklärt werden.

Angenommen drei Teilnehmer wollen genau eine Nachricht anonym verschicken. Dazu tauscht jeder mit jedem eine Zufallszahl aus und speichert die negierte Zahl seines Partners als seinen Schlüssel für diesen Partner. Abbildung 1 zeigt diesen Schritt.

Angenommen Alice will eine Nachricht senden, Bob und Carol decken sie. Bob und Carol senden

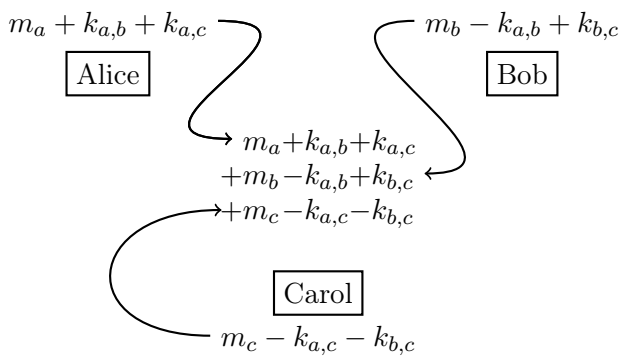


Abbildung 2: Nachrichtenaustausch im DC-Netz

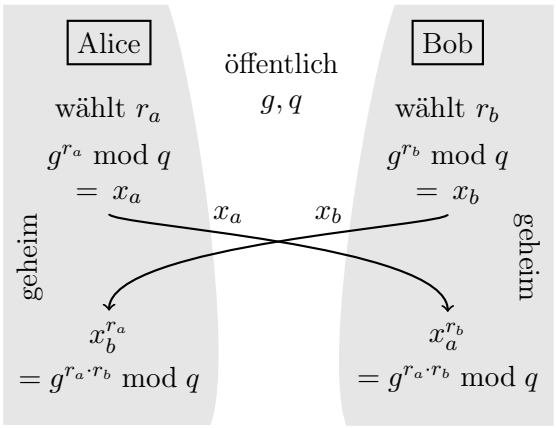


Abbildung 3: Diffie–Hellman Schlüsselvereinbarung

als Nachricht eine 0, Alice die eigentliche Nachricht. Jeder Teilnehmer verschlüsselt seine Nachricht indem er seine Schlüssel die Nachricht addiert.[1] Anschließend wird diese verschlüsselte Nachricht zu einem zentralen Server gesendet.[2] Anzumerken ist, dass niemand ohne Kenntnis der Schlüssel sehen kann, wer was gesendet hat. Werden anschließend alle Nachrichten addiert (Abbildung 2), so heben sich die paarweise inversen Schlüssel auf und das Ergebnis entspricht der Summe aller Nachrichten. Da Bob und Carol eine 0 gesendet haben, entspricht die Summe der Nachrichten Alices' Nachricht.

Würden Bob und Carol statt eine 0 eine andere Nachricht senden, so kann man auf diesem Weg die Summe der Nachrichten aller Teilnehmer berechnet, ohne zu erfahren, wer welche Nachricht geschickt hat. Diesen Effekt macht sich das hier vorgestellte Terminplanungsprotokoll zu nutze.

2.2 Diffie–Hellman Schlüsselvereinbarung

Die Sicherheit der Diffie–Hellman Schlüsselvereinbarung [3] beruht auf der diskreten Logarithmus-Annahme. Die diskrete Logarithmus-Annahme besagt, dass es schwer ist, den Logarithmus in einer diskreten Gruppe zu ziehen. Genauer: Der Logarithmus ist definiert mit $r = \log_g x$, wenn $x = g^r$. Der diskrete Logarithmus ist in einer Gruppe definiert mit $r \equiv \log_g x$, wenn $x = g^r \bmod q$. Die diskrete Logarithmusannahme besagt nun, dass es selbst mit Kenntnis von x, g und q schwer ist r zu berechnen, also den diskreten Logarithmus aus x zu ziehen.[3]

Diese Annahme macht sich die Diffie–Hellmann Schlüsselvereinbarung zu nutze. Ziel ist es, sich über einen unsicheren Kanal auf einen geheimen Schlüssel zu einigen. Angenommen zwei Personen (Alice und Bob) wollen sich auf einen Schlüssel einigen. Sie einigen sich zunächst auf die Parameter g und q.[4] Anschließend wählt jeder eine Zufallszahl (Alice: r_a, Bob: r_b), berechnen $x_a = g^{r_a} \bmod q$ bzw. $x_b = g^{r_b} \bmod q$ und tauschen diese zwei Zahlen (über die ggf. unsichere Leitung) aus.[5] Da laut den Exponentationsgesetzen (alles mod q)

$$x_a^{r_b} = (g^{r_a})^{r_b} = g^{r_a \cdot r_b} = g^{r_b \cdot r_a} = (g^{r_b})^{r_a} = x_b^{r_a}$$

gilt, können beide Seiten einen gemeinsamen Schlüssel ($g^{r_a \cdot r_b}$) berechnen, ohne dass ein Angreifer ihn berechnen kann.[6] Abbildung 3 stellt das Protokoll nochmals schematisch dar.

3 Terminplanungsprotokoll

Im Folgenden wird das Protokoll erklärt, welches eine anonyme Terminplanung leistet. Genereller betrachtet handelt es sich bei dem Protokoll um eine anonyme Abstimmung über verschiedene Zeitpunkte. Eine Abstimmung gliedert sich in 3 Phasen, das Anlegen einer Abstimmung, die Abstimmung selbst und die Ergebnisveröffentlichung. Zusätzlich muss sich jeder potenzielle Teilnehmer einmalig bei einem Schlüsselserver registrieren.

[1] Diese Darstellung ist etwas vereinfacht, die Addition findet normalerweise in einer Restklasse statt.

[2] Statt einen zentralen Server zu benutzen können die Nachrichten auch an alle Teilnehmer verteilt werden.

[3] „Schwer" gilt hier natürlich nur, für einen ausreichen großen Wertebereich.

[4] g und q sind öffentlich

[5] anschließend sind auch x_a und x_b öffentlich

[6] Der Angreifer kennt g, q, x_a, x_b, aber nicht r_a und r_b

3.1 Registrierung

Zur Registrierung generieren alle potenziellen Teilnehmer wie folgt ein asymmetrisches Diffie–Hellman Schlüsselpaar:[7]

1. Generiere eine Zufallszahl r und speichere sie als geheimen Schlüssel.
2. Berechne den öffentlichen Schlüssel $g^r \bmod q$ und veröffentliche ihn.[8]

An dieser Stelle ist ersichtlich, dass jeder Teilnehmer nun die Möglichkeit hat, sich mit Hilfe des öffentlichen Schlüssels eines anderen Teilnehmers auf einen gemeinsamen Schlüssel zu einigen, ohne mit ihm eine Kommunikationsbeziehung aufzubauen. Benötigt Alice (geheimer Schlüssel: r_a) beispielsweise einen gemeinsamen Schlüssel mit Bob (öffentlicher Schlüssel: $g^{r_b} \bmod q$), so berechnet sie $(g^{r_b})^{r_a} \bmod q$. Durch den Umstand, dass g und q für alle Teilnehmer gleich sind, kann jeder über den öffentlichen Schlüssel des anderen einen solchen gemeinsamen geheimen Schlüssel berechnen.

3.2 Anlegen einer Abstimmung

Das Anlegen einer Abstimmung funktioniert genauso wie man es von ähnlichen Web 2.0-Anwendungen kennt. Ein Initiator konfiguriert eine Menge von möglichen Startzeitpunkten für den Termin und versendet sie an die potenziellen Teilnehmer mit der Bitte, ihre Verfügbarkeit zu äußern. Ein Unterschied zu anderen Anwendungen besteht darin, dass der Initiator beim Anlegen der Abstimmung die Menge der Teilnehmer festlegen muss. Damit wird gleichzeitig eine Anonymitätsmenge festgelegt, in der ein Abstimmender agiert.

3.3 Senden der Stimme

Jeder Teilnehmer bildet als erstes seinen unverschlüsselten Verfügbarkeitsvektor. Dieser Vektor enthält für jeden zur Wahl stehenden Zeitpunkt ein Element. Dieses Element ist 0, sollte der Teilnehmer zum angegebenen Zeitpunkt nicht verfügbar sein und 1, wenn er verfügbar ist.

Anschließend wird für jeden anderen Teilnehmer ein gemeinsamer geheimer Schlüssel berechnet.[9] Diese Schlüsselberechnung folgt obigem Schema (vgl. Abschnitt 2.2 und 3.1) und stellt darüber hinaus noch sicher, dass die Schlüssel jedes Teilnehmerpärchens invers zueinander sind.[10]

Abschließend wird der Schlüssel auf den Verfügbarkeitsvektor addiert, womit eine Verschlüsselung erreicht wird wie es schon in Abschnitt 2.1 beschrieben wurde. Dieser verschlüsselte Verfügbarkeitsvektor wird zum Server übertragen.

3.4 Ergebnisveröffentlichung

Haben alle Teilnehmer ihre Stimme abgegeben, so können alle verschlüsselten Verfügbarkeitsvektoren veröffentlicht werden. Addiert man alle Vektoren, so ergibt sich durch die Tatsache, dass alle Schlüssel wegfallen die Summe aller unverschlüsselten Vektoren. Anhand dieses Ergebnisses kann das Datum gewählt werden, an dem der Termin angesetzt wird.

3.5 Verhindern von Angriffen

Um zu verhindern, dass außenstehende Personen angreifen, reicht es wenn jeder seine Nachrichten digital signiert und die Teilnehmer die Signaturen prüfen. Betrachtet man jedoch auch Teilnehmer als Angreifer, so können sie in ihrer Stimme statt 0 oder 1 einen völlig anderen Wert senden (z. B. $+2$ oder -1). Besonders problematisch ist, dass dieser Angriff vollständig anonym stattfinden kann, weil niemand die Einzelstimmen sehen kann. Im Folgenden wird eine Methode vorgestellt, wie diese Angriffe vermieden werden können.

Versucht ein Angreifer eine -1 zu senden, so kann er dies nur dann unbemerkt tun, wenn er davon ausgeht, dass mindestens ein anderer Teilnehmer eine 1 gesendet hat. Das ist in Abbildung 4 dargestellt (zu sehen sind die unverschlüsselten Verfügbarkeitsvektoren). Hier versucht Mallory das Ergebnis zu fälschen, indem sie zu jedem zur Abstimmung stehenden Zeitpunkt eine -1 sendet. Da aber am dritten Zeitpunkt kein anderer eine 1 gesendet hat, ist die Summe hier -1 und ihr Angriff wird entdeckt.

Natürlich wird so ein Angriff mit steigender Teilnehmerzahl weniger wahrscheinlich entdeckt. Um die Erfolgswahrscheinlichkeit für so einen Angriff dennoch zu reduzieren, teilt jeder Teilnehmer seinen Verfügbarkeitsvektor in viele verschiedene Teilvektoren. Die Summe aller Teilvektoren ergibt hier-

[7] vgl. Abschnitt 2.2

[8] g und q sind für alle Teilnehmer gleich.

[9] Genaugenommen wird für jeden Zeitpunkt und für jeden Teilnehmer ein Schlüssel berechnet.

[10] z. B. dadurch, dass derjenige, dessen öffentlicher Schlüssel kleiner ist sein Ergebnis invertiert

	T1	T2	T3	T4
Alice	1	1	0	0
Bob	0	1	0	1
Mallory	−1	−1	−1	−1
$\sum$	0	1	−1	0

Abbildung 4: Naiver Angriff von Mallory durch Senden des Wertes −1 zu jedem Zeitpunkt. Dargestellt sind die unverschlüsselten Verfügbarkeitsvektoren. Bei Zeitpunkt T3 fällt auf, dass die Summe außerhalb des zulässigen Wertebereiches liegt.

bei den Verfügbarkeitsvektor. Anschließend wird obiges Protokoll für jeden Teilvektor einzeln ausgeführt. Dies ist nochmals in Abbildung 5 dargestellt. Hier sieht man, dass Mallory in der linken Seite mit ihrem Angriff durchkommt. Rechts muss sie jedoch raten, an welcher Stelle Alice und Bob ihre Einsen gesendet haben. Ihr Angriff wird dadurch mit höherer Wahrscheinlichkeit erkannt.

Auf die gleiche Weise, wie „−1-Angriffe“ erkannt werden können, können Angriffe mit +2 erkannt werden. Dazu invertiert jeder Teilnehmer seinen Vektor und führt das gleiche Protokoll nochmals auf den invertierten Vektor aus. Da die Summe beider Abstimmungen (invertierte und nicht invertierte) in jeder Spalte der Summe der Teilnehmer gleichen muss, muss ein Angreifer für eine +2 in der einen Tabelle auch eine −1 in der invertierten Tabelle senden, was mit dem gerade erklärten Protokoll erschwert wird. Abbildung 6 stellt das ganze nochmals dar.

4 Zusammenfassung

Es wurde ein Protokoll vorgestellt, welches es ermöglicht, Termine zu planen ohne das gesamte Verfügbarkeitsmuster aufzudecken. Darüberhinaus eignet es sich um kleinere Umfragen zu gestalten. Im Gegensatz zu anderen E-Voting-Verfahren beruht das Vertrauensmodell hier nicht auf zentralen Entitäten, sondern hängt an der Geheimhaltung der Schlüssel jedes Teilnehmers. Durch diesen Umstand skaliert das Protokoll nicht mehr in der Dimension der Teilnehmer. Das stellt aber für die konkrete Anwendung kein Problem dar, weil es nicht darum geht, ein Wahlprotokoll für mehrere Millionen Bürger zu erstellen, sondern ein Protokoll in einer kleinen abgeschlossenen Gruppe von Personen. Das Protokoll wurde hier vereinfacht vorgestellt. Leser, die an einer ausführlicheren Beschreibung interessiert sind, können einen ausführlicheren Beitrag in den Proceedings der PASSAT '09 nachlesen [4].

Das Protokoll wurde schon teilweise in Javascript implementiert [5]. Die Implementierung enthält zur Zeit noch nicht alle Features, wird aber in den kommenden Monaten weiter ausgebaut.

Danksagung

Der Autor möchte sich für die anregenden Diskussionen und Kommentare bei Josef Schmid und der gesamten Dresdner DuD-Gruppe bedanken. Diese Arbeit wurde von dem European Community's Seventh Framework Programme (FP7/2007–2013) Bewilligung № 216483 gefördert.

Literatur

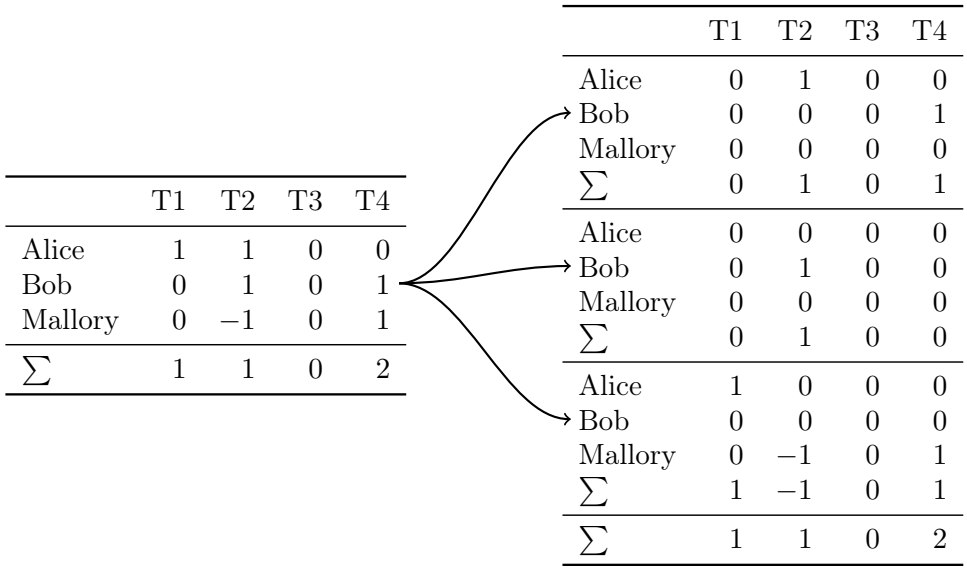
[1] Näf, Michael: *Doodle Homepage.* `http://www.doodle.com`, Dezember 2009.

[2] Chaum, David: *The dining cryptographers problem: Unconditional sender and recipient untraceability.* Journal of Cryptology, 1(1):65–75, Januar 1988.

[3] Diffie, Whitfield und Martin E. Hellman: *New Directions in Cryptography.* IEEE Transactions on Information Theory, IT-22(6):644–654, 1976.

[4] Kellermann, Benjamin und Rainer Böhme: *Privacy-Enhanced Event Scheduling.* In: *IEEE International Conference on Computational Science and Engineering*, Band 3, Seiten 52–59, Los Alamitos, CA, USA, August 2009. IEEE/IFIP, IEEE Computer Society. Conference on Information Privacy, Security, Risk and Trust (PASSAT '09).

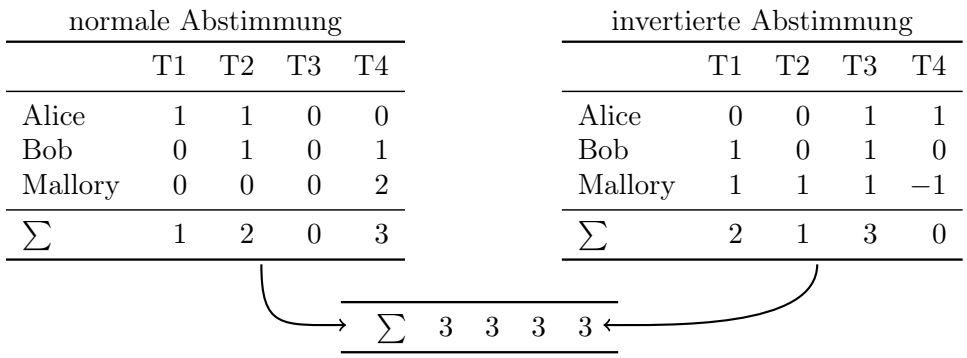
[5] Kellermann, Benjamin: *Dudle Homepage.* `http://dudle.inf.tu-dresden.de`, Dezember 2009.



	T1	T2	T3	T4
Alice	1	1	0	0
Bob	0	1	0	1
Mallory	0	−1	0	1
$\sum$	1	1	0	2

	T1	T2	T3	T4
Alice	0	1	0	0
Bob	0	0	0	1
Mallory	0	0	0	0
$\sum$	0	1	0	1
Alice	0	0	0	0
Bob	0	1	0	0
Mallory	0	0	0	0
$\sum$	0	1	0	0
Alice	1	0	0	0
Bob	0	0	0	0
Mallory	0	−1	0	1
$\sum$	1	−1	0	1
$\sum$	1	1	0	2

Abbildung 5: Aufteilen der Stimme auf verschiedene Vektoren. In der linken Tabelle bleibt Mallory noch unbemerkt, durch aufteilen der Stimmen hat Mallory in der rechten Tabelle nur noch eine Wahrscheinlichkeit von $\frac{2}{3}$ um unbemerkt zu bleiben.



normale Abstimmung

	T1	T2	T3	T4
Alice	1	1	0	0
Bob	0	1	0	1
Mallory	0	0	0	2
$\sum$	1	2	0	3

invertierte Abstimmung

	T1	T2	T3	T4
Alice	0	0	1	1
Bob	1	0	1	0
Mallory	1	1	1	−1
$\sum$	2	1	3	0

$\sum$	3	3	3	3

Abbildung 6: Durch invertieren der Tabelle wird das „+2-Problem“ auf das „−1-Problem“ abgebildet. Mallory muss bei T4 in der 2. Tabelle eine −1 senden, weil sonst die spaltenweise Summe beider Tabellen nicht mehr der Anzahl der Teilnehmer gleicht.