Key Agreement

Hier geht es um die Berechnung eines gemeinsamen Geheimnisses durch A und B, welches zur Erzeugung symmetrischer Schlüssel verwendet werden kann.

A und B führen dazu ein Key Agreement Protokoll aus.

Erforderliche bzw. wünschenswerte Eigenschaften:

- Authenticated Key Agreement (with Key Confirmation)
- Known-key Security, Forward Secrecy, Key Compromise Impersonation, Unknown Key-Share.

Um diese Eigenschaften zu erreichen, verwenden A und B öffentliche/private Schlüssel.

Key Agreement

Man geht davon aus, daß A den authentischen öffentlichen Schlüssel von B hat und umgekehrt. Sonst ist prinzipiell ein Man-in-the-Middle Angriff möglich.

Die Annahme kann durch Key Management Techniken erreicht werden, wie zum Beispiel durch Zertifikate einer übergeordneten Zertifizierungsbehörde (CA).

Zertifikate sind digitale Signaturen der öffentlichen Schlüssel unter dem öffentlichen Schlüssel der CA. Damit wird das Problem der Authentifizierung im Prinzip nur verlagert, in der Praxis aber erreicht.

Die öffentlichen Schlüssel der CA sind üblicherweise in Webbrowsern „fest“ eingebaut. Man-in-the-Middle Angriffe durch Zwischenstationen im Internet werden damit verhindert.

Authentifizierter Diffie-Hellman Schlüsselaustausch

1. A schickt B eine Nonce N_A.
2. B wählt ein zufälliges r_B und schickt r_BP an A.
3. A wählt ein zufälliges r_A und schickt r_AP an B.
4. A und B berechnen unabhängig $r_A(r_BP) = r_B(r_AP) = r_Ar_BP$ als gemeinsames Geheimnis.

A und B schicken in Schritt 2 und 3 zusätzlich ihre Unterschriften über die bisher ausgetauschten Werte mit bzw. überprüfen diese in Schritt 3 und 4.

Die Nonce dient dazu, A zu versichern, daß B selbst antwortet (kein replay). Die umgekehrte Rolle wird von r_B gespielt.

MQV Protokoll

MQV Protokolle:

- Schlüsselberechnung symmetrisch.
- w_A, W_A, w_B, W_B private/öffentliche Schlüssel.
- Geheime Zufallszahlen r_A, r_B. $R_A = r_AP$ und $R_B = r_BP$ werden von A und B ausgetauscht.
- A berechnet $s_A = (r_A + \mu(R_A)w_A) \bmod \ell$, $S_B = (R_B + \mu(R_B)W_B)$ und $K_{AB} = hs_AS_B$. B analog mit A und B vertauscht.

Two-Pass Variante: AK, keine Key Confirmation.

One-Pass Variante von A nach B: Verwendet w_B, W_B anstelle von r_B, R_B. AK, keine Key Confirmation. Keine known-key oder forward secrecy, da B passiv ist.

Unknown Key-Share möglich (durch Zusatz leicht verhinderbar).

MQV Protokoll

MQV Three-Pass Protokoll:

- Rechnungen sind wie in Two-Pass Protokoll, plus folgende, zusätzliche Schritte zwischen den Datenübertragungen.
- Seien H_1, H_2 unabhängige Hashfunktionen ($H_i(x)$ = SHA-1($i||x$)).
- A sendet R_A.
- B berechnet den Session Key $k = H_1(K_{BA})$ und Authentifizierungsschlüssel $k' = H_2(K_{BA})$.
- B sendet R_B und MAC$_{k'}(2, B, A, R_B, R_A)$.
- A berechnet den Session Key $k = H_1(K_{AB})$ und Authentifizierungsschlüssel $k' = H_2(K_{AB})$ und überprüft den MAC-Wert.
- A sendet MAC$_{k'}(3, A, B, R_A, R_B)$, und B überprüft den MAC-Wert.

MQV Protokoll

Eigenschaften:

- AK mit Key Confirmation, hat alle oben genannten Sicherheitsmerkmale (allerdings heuristisch, ohne Beweis).
- Effizient.

Bemerkung:

- Entwurf von (effizienten) Key Agreement Protokollen und anderen höher stehenden Protokollen sehr trickreich.
- MQV und unknown key-share Angriff verdeutlichen dies ...

Salzen und Strecken

Nochmal zu den Paßwörtern. Im folgenden zwei Techniken, einen Dictionary-Angriff zu erschweren, sollte der Angreifer Zugriff auf die Paßwortdatei mit Einträgen (ID$_i$, H(Passwort$_i$)) haben.

Strecken:

- Statt H einmal anwenden, H sehr häufig anwenden, also Paßwortdatei mit Einträgen der Form (ID$_i$, H^r(Passwort$_i$)) verwenden.
- r sollte so groß sein, daß die Berechnung beispielsweise ca. 1 Sekunde dauert.
- Der Angreifer hat dann einen unumgänglich größeren Rechenaufwand.

Salzen und Strecken

Salzen:

- Zu den Paßwörtern Salz hinzufügen, also Paßwortdatei mit Einträgen (ID$_i$, H^r(Passwort$_i$||Salz$_i$), Salz$_i$) verwenden.
- Zum Beispiel Salz$_i = i$, oder besser Salz$_i$ ein Zufallswert.
- Dictionary-Angriff bezieht sich dann nur auf eine gewählte ID$_i$, nicht alle zusammen (beachte Geburtstagsparadxon), bzw. wird für alle erschwert.
- Ist Paßwortdatei nicht lesbar, dann online Angriff auf feste ID$_i$ zusätzlich schwerer.
- Login Prompt hat normalerweise eingebaute Verzögerung, wenn Paßwort falsch eingegeben wird.

Standards

Ziel: Interoperabilität, sichere Implementierungen.

Zu definieren:

- Format der Daten, Körperelemente, Punkte, Signaturen, (Public-Key) Chiffretexte, ...
- Konversionsfunktionen dieser Objekte nach Bitstring, Integers, ...
- Genauer Spezifikation der Algorithmen, Punktkompression ja/nein, Hybridverschlüsselung, ...

Ein paar Standards/Organisationen:

- International: IEEE, IETF, ISO, ITU.
- National: ANSI, CEN, DIN, NIST.
- Company: PKCS, SECG.

Standards

IEEE P1363/P1363a: Enthält alle Public-Key Verfahren.

- EC-DH, EC-DSA, EC-MQV, EC-IES.
- Anhang enthält viele grundlegende, zahlentheoretische Algorithmen.

ANSI X9.62: EC-DSA.

ANSI X9.63: EC-DH, EC-MQV, EC-IES.

FIPS186.2: NIST Standard für DSA und EC-DSA.

SECG: Indutriestandard geführt von Certicom.

- EC-DH, EC-DSA, EC-MQV, EC-IES.

IETF: IPsec, S/MIME, ...

ISO/IEC 14888-3: Identity-based cryptography.

PKCS #13: ECC.

Standards

Alle (außer IEEE) geben Nachrichtenformate und empfohlene Kurven an.

- Nicht jeder kann/will geeignete Körper und Kurven selber finden (Punktezählalgorithmus implementieren, ...)
- Unterstützt Verbreitung und Interoperabilität.

Verschiedene Standards benutzen teilweise die gleichen Kurven.

- Unterstützt Verbreitung und Interoperabilität.

Die ANSI und NIST Kurven sind „am besten“.

Arbeitsweise von Standards:

- „Freiwillige“ Mitarbeit von Firmenangestellten. Wenig Academia.
- Dadurch Interessensvertretung der Firmen ...

Interaktive Beweissysteme

Hier nur informell.

Haben zwei Programme, einen Beweiser P und einen Verifizierer V. Diese tauschen Nachrichten aus, zum Schluß akzeptiert V oder weist zurück.

P und V haben als Eingabe jeweils ein Zufallsband und weitere, gemeinsame Daten. P erhält darüberhinaus weitere (vor V geheimen) Daten, welche Zeuge genannt werden.

Die Idee ist, daß P die Lösung zu einem Problem erhält und V davon überzeugen muß, daß es die Lösung kennt, ohne etwas über die Lösung zu verraten.

Allgemein betrachten wir (P,V) als Programm mit zwei kommunizierenden, beliebigen Teilprogrammen P und V.

Interaktive Beweissysteme

Beispiel: Für ein y kennt P ein x mit $y = g^x$. V kennt nur y. P will beweisen, daß es x kennt.

- P könnte x an V schicken, aber dann wäre es V bekannt.
- Abgesehen davon gibt es Probleme und Lösungen x, wo es schwierig ist, zu prüfen, ob x eine Lösung ist. Betrachten wir hier nicht.

Es sollen folgende Eigenschaften gelten:

- Vollständigkeit: Wenn P den Zeugen x als Eingabe erhält, dann akzeptiert V.
- Korrektheit: Für jedes Programm B: Wenn V in der Interaktion mit B akzeptiert, dann hat B den Zeugen mit Wahrscheinlichkeit $\geq 1/3$ als Eingabe erhalten.

Dann heißt (P,V) ein interaktives Beweissystem.

Zero Knowledge Beweise

Daß P nichts weiteres bezüglich x verrät, wird wie folgt formalisiert.

Für jedes Programm V^* gibt es ein Programm M^*, dessen Ausgaben als Zufallsvariable mit den Ausgaben von (P,V^*) als Zufallsvariable in Abhängigkeit der Zufallsbänder nicht effizient unterschieden werden können (bei gleicher, fixer Eingabe der zusätzlichen Parameter, bis auf den Zeugen).

Die Eingabe des Zeugen soll also für eine (jede) Berechnung keinen Vorteil bringen.

Es genügt, nur solche M^* zu betrachten, welche die zwischen P und V^* kommunizierten Nachrichten und das Zufallsband von V^* ausgeben, da diese als Eingabe von V^* aufgefaßt werden können und V^* danach deterministisch abläuft. Man nennt M^* einen Simulator.

Zero Knowledge Beweise

Sei n RSA Modul, x zufällig und $y = x^2 \bmod n$.
P soll Kenntnis von x beweisen.

1. P wählt zufälliges r und schickt $a = r^2$ an V.
2. V wählt zufälliges $e \in \{0,1\}$ und schickt e an P.
3. P berechnet $b = rx^e$ und schickt b an V.
4. V akzeptiert genau dann, wenn $b^2 = ay^e$.

Ein schummelndes Programm B anstelle von P kann wie folgt mit Erfolgswahrscheinlichkeit $1/2$ vorgehen:

1. B wählt zufälliges r und rät e im voraus, also $e' \in \{0,1\}$ zufällig.
2. B berechnet $a = r^2 y^{-e'}$ und schickt a an V.
3. V schickt zufälliges e an B, und B antwortet mit r.

Zero Knowledge Beweise

Angenommen, es gibt ein schummelndes B mit Erfolgswahrscheinlichkeit $> 1/2$.

Dann kann B Wurzeln b, r mit $b^2 = ay^e$ und $r^2 = ay^{e'}$ und $e \neq e'$ finden. Folglich kann B Quadratwurzeln von y ausrechnen, also n faktorisieren.

Durch mehrfache Wiederholung des interaktiven Beweises kann man die Erfolgswahrscheinlichkeit für B beliebig klein machen.

Zero Knowledge Beweise

Simulation des Beweises: Sei V^* gegeben. Wir konstruieren M^* wie B, nur daß M^* noch $e' = V^*(a)$ überprüft und solange rechnet, bis dies gilt.

Die Ausgabe von M^* ist dann nicht von der von (P, V^*) zu unterscheiden:

- a ist zufällig, $e = V^*(a)$, und b ist eine korrekte Quadratwurzel.

Damit ist der angegebene interaktive Beweis ein Zero Knowledge Beweis (of Knowledge).

Commitment Schemes

Ziel: P legt sich vor V auf einen Wert fest, ohne daß V den Wert erfährt. Später muß P den Wert bekannt geben.

Eigenschaften:

- Hiding (comput., uncond.): Der Wert ist geheim vor V.
- Binding (comput., uncond.): P kann bei Bekanntgabe nicht schummeln.

Variante basierend auf DLP:

1. Challenge. V schickt g, v an P.
2. Commit. P legt sich auf die Nachricht m wie folgt fest. P wählt zufälliges r und schickt $c = g^r v^m$ an V.
3. Reveal. P schickt m, r an V. V testet $c = g^r v^m$.

Commitment Schemes

Wenn P schummeln könnte, dann hätten wird eine Gleichung $g^r v^m = g^{r'} v^{m'}$ und somit das DLP $v = g^{(r-r')/(m'-m)}$.
P kann $(r - r')/(m' - m)$ berechnen.

Der Wert c ist völlig zufällig, gibt keine Information über m preis.
V hat daher auch dann keinen Vorteil, wenn es s mit $v = g^s$ kennt.

Beispiel: Münzwurf übers Telefon.
Scheme ist unconditionally hiding und computationally binding.

Außerdem:
Mit $C(m, r) = g^r v^m$ gilt $C(r_1, m_1)C(r_2, m_2) = C(r_1 + r_2, m_1 + m_2)$.
$\Rightarrow$ homomorphes Commitment.

Dies findet Anwendung beim verdeckten Rechnen, e-Voting.

Weitere Protokolle

- Multiparty Kryptographie.
- Threshold Kryptographie („Entschlüsseln, wenn die Mehrheit dafür ist“).
- Multi-party computation.

- Oblivious transfer, simulaneous contract signing.
- Auktionen.
- ... ...
- ...