Digitale Signaturen

Signaturverfahren mit Einwegfunktion mit Falltür:
- Full Domain Hash, RSA Signatures, PSS

Signaturverfahren mit Einwegfunktion ohne Falltür:
- Allgemeine Konstruktion von Lamport, One-time Signatures, Authentication tree.

Signaturverfahren mit Einwegmonomorphismus:
- Schnorr Signaturen (beweisbar sicher im RO, Reduktion zum DLP), DSA.

Spezielle Signaturverfahren:
- Blinde Signaturen, Undeniable Signatures.

Diffie-Hellman Schlüsselaustausch

Ziel: A und B wollen (mithilfe von öffentlichen Schlüsseln, Master Keys) einen gemeinsamen, geheimen Schlüssel (Session Key) bestimmen. Dieser kann dann beispielsweise für effiziente symmetrische Verfahren genutzt werden.

Sei $(G,+)$ eine Gruppe von Primzahlordnung ℓ mit Erzeuger P.

1. A wählt ein zufälliges $r_A \in \mathbb{Z}/\ell\mathbb{Z}$ und schickt $r_A P$ an B.
2. B wählt ein zufälliges $r_B \in \mathbb{Z}/\ell\mathbb{Z}$ und schickt $r_B P$ an A.
3. A berechnet $r_A(r_B P) = r_A r_B P$. B berechnet $r_B(r_A P) = r_A r_B P$.

Will ein passiver Angreifer das gemeinsame Geheimnis erfahren, muss er das CDH zu P, $r_A P$, $r_B P$ lösen.

Diffie-Hellman Schlüsselaustausch

Ein aktiver Angreifer kann leicht einen Man-in-the-Middle Angriff starten (woher weiß A, daß $r_B P$ von B kommt?), daher in dieser Form unsicher.

Also müssen die Nachrichten $r_A P$ und $r_B P$ authentifiziert werden. Dies kann beispielsweise mit einer Unterschrift von A bzw. B erfolgen.

Nun Man-in-the-Middle Angriff auf die öffentlichen Schlüssel (woher weiß A, daß der öffentliche Schlüssel von B wirklich von B ist?).

Zertifikate verwenden: Eine vertrauenswürdige Zertifizierungsstelle (CA) überprüft die Authentizität (Name, Adresse) der öffentlichen Schlüssel und bestätigt dies mit ihrer Unterschrift (Zertifikat).

Der authentische, öffentliche Schlüssel der CA ist allseits bekannt ...
Verschiedene CA zertifizieren ihre Schlüssel gegenseitig ...

Key Management und Establishment

A und B können also die Authentizität der jeweiligen öffentlichen Schlüssel mit Hilfer der Zertifikate überprüfen. Dieser Schritt fällt in den Bereich des Key Managements.
- zum Beispiel weitere Funktion: Zertifikatsrückruflisten (CRL).

Hier jetzt nur Key Establishment. Unterteilt sich in
- Key Transport: A erzeugt Schlüssel und transportiert ihn zu B.
- Key Agreement: A und B berechnen gemeinsam den Schlüssel, Ergebnis nicht vorhersehbar.

Beide Techniken gibt es für symmetrische und asymmetrische Verfahren (z.B. KT symm: Kerberos, KT asymm: X.509, siehe HAC). Betrachten im folgenden nur Key Agreement mit asymmetrischen Verfahren.

Key Agreement

Hier geht es also 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 (s.u.):

- 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.

Diffie-Hellman Schlüsselaustausch

Zurück zum DH Schlüsselaustausch. In der obigen Form ist ein Replay Angriff möglich, E kann sich zu einem späteren Zeitpunkt als A oder B ausgeben und die mitgehörten Daten senden (aber keinen Session Key erhalten).

Abhilfe: Im nullten Schritt schickt A eine Nonce an B. Die Unterschriften von A und B enthalten dann alle jeweils gesendeten und empfangenen Daten.

Um etwaige schwache Bits und algebraische Eigenschaften zu kaschieren, empfiehlt es sich, im dritten Schritt noch den Hashwert von $r_A r_B P$ für eine geeignete Hashfunktion zu berechnen (z.B. SHA-256, um einen AES Schlüssel zu erhalten).

Mehr Flexibilität: Im nullten Schritt kann man B noch „Vorschläge“ für G schicken lassen, von denen sich B im ersten Schritt eine aussucht und die Parameter an A schickt, welcher diese überprüft.

Authentifizierter Diffie-Hellman Schlüsselaustausch

0. A schickt B eine Nonce N_A.
1. B wählt ein zufälliges r_B und schickt $r_B P$ an A.
2. A wählt ein zufälliges r_A und schickt $r_A P$ an B.
3. A und B berechnen unabhängig $r_A(r_B P) = r_B(r_A P) = r_A r_B P$ als gemeinsames Geheimnis (z.B. SHA-256$(r_A r_B P)$).

A und B schicken in Schritt 1 und 2 zusätzlich ihre Unterschriften über die bisher ausgetauschten Werte mit bzw. überprüfen diese in Schritt 2 und 3.

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

Diffie-Hellman Schlüsselaustausch

Diskussion der Protokolleigenschaften.

B’s Sicht:

- Im nullten Schritt erhält B eine Zahl und vorgeschlagene Parameter, könnte von jedem kommen.
- Im zweiten Schritt erhält B eine authentische Nachricht von A. Diese ist kein Replay, da sie $r_B P$ enthält. Außerdem enthält die die Daten des nullten Schritts, die damit auch authentisch sind.
- G ist sicher nach Wahl durch B. Folglich kann $r_A r_B P$ nur mit Kenntnis von r_A aus $r_B P$ berechnet werden. Durch die Unterschriften „bestätigt“ A die Kenntnis von r_A.
- Folglich kann nur A den Wert $r_A r_B P$ berechnen.

Diffie-Hellman Schlüsselaustausch

A’s Sicht:

- Im ersten Schritt erhält A eine authentische Nachricht von B. Diese ist kein Replay, da sie Nonce von A enthält.
- A überprüft G auf Sicherheit (sinnvolle, korrekte Parameter). Daher kann $r_B r_A P$ aus $r_A P$ nur mit Kenntnis von r_B bestimmt werden. Durch die Unterschrift „bestätigt“ B die Kenntnis von r_B.

E’s Sicht:

- Passiver Angriff hilft nicht, da G sicher. Also nur aktive Angriffe.
- Daten abändern geht nicht wegen der letzten Unterschrift.
- Replay geht nicht wegen der Nonce und $r_A P$.

A’s privater Schlüssel zum Signieren wird öffentlich: Alte Session Keys bleiben sicher. A’s und B’s aktueller Session Key wird öffentlich: Keine Auswirkung auf die anderen Schlüssel.

Terminologie, Ziele

- Implicit Key Authentication (von B nach A): A ist sicher, daß nur B Session Key erhalten kann.
- Authenticated Key Agreement (AK): Implicit Key Authentication in beide Richtungen.
- Key Confirmation (von B nach A): A ist sicher, daß B den Session Key besitzt.
- Explicit Key Authentication: Implicit Key Authentication + Key Confirmation.
- Authenticated Key Agreement with Key Confirmation (AKC): Explicit Key Authentication in beide Richtungen.

Key Confirmation kann durch Ver-/Entschlüsseln geeigneter Testnachrichten nach Schlüsselaustausch erfolgen.

Eigenschaften

Sicherheit:

- Known-key Security: Protokoll funktioniert korrekt, auch wenn alte oder der aktuelle Session Key bekannt wird.
- Forward Secrecy: Alte Session Keys bleiben sicher, auch wenn Master Keys bekannt werden.
- Key Compromise Impersonation: Bekannter Master Key von A führt nicht zur Impersonation Angriff gegenüber A.
- Unkown Key-Share: Es ist nicht möglich, daß A denkt, er teilt Session Key mit C $\neq$ B, wenn Session Key mit B geteilt wird.
- Key Control: Weder A noch B können Session Key wählen.

Eigenschaften, Aufgaben

Performance, wünschenswert:

- Wenig Runden, effiziente Ausführung, symmetrisch.
- Nicht interaktiv, keine Verschlüsselung, keine Hashfunktionen, ...

Key Derivation Function: Session Keys aus gemeinsamen Geheimnis konstruieren.

Domain Parameter Validation: Überprüfen, daß G bzw. die Parameter von sicher sind.

Key Validation: Überprüfen, daß Master Key und Session Keys sicher sind (Erzeuger der zyklischen Gruppe mit großem Primfaktor).
Ersteres kann man der CA und dem Zertifikat überlassen.

Embedded Key Validation: Key Validation geschieht implizit im Key Agreement Protokoll.

Diffie-Hellman Schlüsselaustausch

Der authentifizierte Diffie-Hellman Schlüsselaustausch hat offenbar die folgenden Eigenschaften:

- Ist Authenticated Key Agreement, Key Confirmation nur „50%“.
- Known-key Security, Forward Secrecy, Key Compromise Impersonation, Unkown Key-Share, Key Control.
- Domain Parameter und Key Validation müssen durchgeführt werden.

Im folgenden: MQV Protokolle, nach Menezes-Qu-Vanstone (1995).

- Effizienter als authentifizierter DH Key Exchange wie oben.
- Rechte an EC-MQV sind 2003 an NSA für 25 Mill. Dollar verkauft worden (inklusive Techniken/Benutzung von Patenten für ECC).
- Auch im IEEE P1363 Standard.

MQV Protokoll

Sei $\mu : G \to [2^f, 2^{f+1} - 1]$ eine (im wesentlichen) injektive Funktion mit $f = \lfloor \log_2(\ell) \rfloor + 1$. Der Kofaktor sei $h = \#G/\ell$.
$w_A, W_A = w_A P$, $w_B, W_B = w_B P$ private/öffentliche Schlüssel von A und B.

MQV Two-Pass Protokoll: Ist symmetrisch in A und B, beschreiben daher nur eine Hälfte.

- A erzeugt zufälliges r_A mit $1 \leq r_A \leq \ell - 1$ und schickt $R_A = r_A P$ an B (zusammen mit einem Zertifikat für A’s Masterkey).
- A überprüft das Zertifikat von B und $R_B \in G$.
- A berechnet $s_A = (r_A + \mu(R_A)\mu(W_A) w_A) \bmod \ell$, $S_B = (R_B + \mu(R_B)\mu(W_B) W_B)$ und $K_{AB} = h s_A S_B$.
- A überprüft $K_{AB} \neq O$, sonst Abbruch.

Es gilt $K_{AB} = K_{BA}$.

MQV Protokoll

Eigenschaften:

- Ist AK Protokoll, keine Key Confirmation.
- Hat alle oben aufgeführten Sicherheits- und Performanceeigenschaften (ohne Beweis allerdings).
- K_{AB} sollte durch Multiexponentiation berechnet werden.

Haben 1995 Variante beschrieben, da 1998 Variante einen Unkown Key-Share Angriff erlaubt. Letztere erhält man, wenn man die blauen Terme fortläßt.

Unkown Key-Share Angriff: In Abhängigkeit von R_A kann E die Schlüssel $w_E, W_E = w_E P$ und R_E so erzeugen, daß $S_E = R_E + \mu(R_E) W_E = S_A$ gilt ($R_E = S_A - \lambda P$, $w_E = \lambda \mu(R_E)^{-1} \bmod \ell$).
Der Session Key ist dann sowohl für E, A als auch für B, A gültig.

MQV Protokoll

Maßnahmen gegen den Unknown Key-Share Angriff:

- Key Confirmation.
- Key Derivation in Abhängigkeit der öffentlichen Schlüssel.

MQV One-Pass Protokoll:

- Wie Two-Pass Protokoll ohne blaue Terme, auf A's Seite statt R_B den Schlüssel W_B und auf B's Seite w_B bzw. W_B statt r_B bzw. R_B verwenden.
- Nützlich, wenn B nicht online.

Sicherheitseigenschaften:

- Ist AK Protokoll, keine Key Confirmation.
- Keine known-key security oder forward secrecy, da B passiv ist.

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 ...