Authenticated Encryption (1)

Krypto II

VL 18

03.12.2018

Buch des Tages

Titel: Modern Cryptography and Elliptic Curves

Autor(en): Thomas Shemanske

Verlag: AMS American Mathematical Society, 2017

Umfang: ca. 245 Seiten

Hinweise zum Inhalt:

Das Buch ist eine Einführung in die Elliptic Curve Cryptography. Der Text ist als Einführung in die Thematik konzipiert und liefert daher zugleich die mathematischen Grundlagen zum Verständnis elliptischer Kurven. Etwa zwei Drittel des Buches nimmt dieser hinführende Teil ein. Dort finden sich noch andere kurze Abschnitte, die nichts mit elliptischen Kurven zu tun haben und eher in eine allgemeine Einführung in die Kryptographie gehören.

Da es in einer Undergraduate Einführung in Kryptographie nicht immer möglich ist, die benötigte Mathematik für ellipt. Kurven zu erarbeiten, dient dieses Buch als Ergänzung zum Selbstlernen für alle, die sich in die Thematik einarbeiten möchten.

Datenintegrität und Authentizität

- Wir haben uns ausführlich beschäftigt mit Verschlüsselungsmethoden zum Schutz der Vertraulichkeit.
- Bei Betrachtung von Betriebsmodi von Blockchiffren konnten wir feststellen, dass die Integrität der Nachricht nicht immer ausreichend geschützt ist, wenn ein Angreifer beispielsweise Datenblöcke vertauschen, löschen, einfügen oder erneut einspielen (replay) kann.
- Die Urheberschaft bzw. Identität des Absenders ist ebenfalls nicht automatisch gesichert. Bei Public Key Verfahren kann jeder eine Nachricht verschlüsseln und an den Empfänger senden. Bei Secret Key Verfahren kann das zunächst nur eine Person, die über den geheimen Schlüssel verfügt. Dennoch ist auch hier Betrug möglich:
  - Wiedereinspielen früherer Nachrichten durch Dritte
  - Komposition verschlüsselter Nachrichtenteile zu einer neuen Nachricht
  - Verfassen einer Nachricht durch den Empfänger selbst anstelle des vermeintlichen Absenders

Mechanismen zum Integritätsschutz

Eine Nachricht lässt sich auf vielfältige Weise gegen unbemerkte Veränderungen schützen. Die uns bereits bekannten Mechanismen hierzu sind folgende:

- Digitale Signaturen
- Hashwerte
- Symmetrische Verschlüsselung
- Message Authentication Codes

- Ein ***Message Authentication Code MAC*** ist eine Art kryptographische Prüfsumme (***Cryptographic Checksum***), die an die Nachricht angefügt wird. Dies kann mithilfe digitaler Signaturen (Public Key Verfahren) erfolgen, aber auch mit anderen Methoden.

Reihenfolge der Schutzmaßnahmen

Möchte man sowohl Vertraulichkeit als auch Authentizität einer Nachricht kryptographisch sichern, so gibt es vier grundsätzliche Möglichkeiten:

1. zum einen den Plaintext verschlüsseln und zum anderen für den Plaintext einen MAC erstellen/anfügen, beide Aktionen unabhängig voneinander
2. erst den Plaintext verschlüsseln und danach für den Ciphertext einen MAC erstellen/anfügen
3. erst einen MAC erstellen/anfügen und danach Plaintext und MAC zusammen verschlüsseln
4. Verwendung eines Verfahrens, das beides zugleich realisiert: Verschlüsselung und Message Authentication.

Encrypt-and-MAC (1)

- Im Folgenden bezeichne P den Plaintext, C den Ciphertext, $E_K()$ die Verschlüsselung mit Encryption Key K, $MAC_{K'}()$ einen mittels Authentication Key K erstellten Message Authentication Code.
- Die Variante ***Encrypt-and-MAC*** führt einfach zwei voneinander getrennte Funktionen aus: zum einen wird der Plaintext verschlüsselt, und zum anderen wird für den Plaintext ein MAC erstellt, beide Aktionen unabhängig voneinander.
  - Sender und Empfänger verfügen über die beiden Schlüssel K und K'.
  - Der Sender berechnet $C = E_K(P)$ sowie $T = MAC_{K'}(P)$.
  - Das Ergebnis wird an den Empfänger übermittelt: C, T.
  - Der Empfänger berechnet $P' = E^{-1}_K(C)$. Danach vergleicht er T mit $T' = MAC_{K'}(E^{-1}_K(C))$. Stimmen T und T' überein, so weiß der Empfänger, dass P' = P authentisch ist.

Encrypt-and-MAC (2)

- Wie sicher ist die Variante Encrypt-and-MAC? Ein Kritikpunkt besteht darin, dass – in Abhängigkeit vom verwendeten Message Authentication Code, durch den MAC ein Information Leakage erfolgen könnte.
- Wählt man für den MAC eine Funktion, deren Output die Form eines Pseudozufallsstrings hat, so verhindert man ein Information Leakage.
- Die Variante Encrypt-and-MAC wird beispielsweise verwendet in SSH. Zusätzlich zum Key K und dem Plaintext P benötigt SSH noch eine Paketnummer N, mit deren Hilfe beim Empfänger die Datenpakete wieder in die korrekte Reihenfolge gebracht werden können. SSH bildet also $MAC_K(N \| P)$. Als MAC Algorithmus kommt hierbei z.B. HMAC-SHA-256 zum Einsatz.

MAC-then-Encrypt (1)

- Die Kurzbezeichnung ***MAC-then-Encrypt*** beschreibt das Vorgehen bereits: erst berechnet der Sender für den Plaintext einen MAC und fügt diesen an die Nachricht an. Danach werden dann Plaintext und MAC zusammen verschlüsselt:

  - Der Sender berechnet erst $MAC_{K'}(P)$ und dann den Ciphertext $C = E_K(P \| MAC_{K'}(P))$.
  - Das Ergebnis C wird an den Empfänger übermittelt.
  - Der Empfänger berechnet $E^{-1}_K(C)$. Nach Ermittlung von P und $MAC_{K'}(P)$) überprüft er, ob der MAC zur Nachricht P passt.

- Der Empfänger muss in jedem Fall erst entschlüsseln um feststellen zu können, ob die Nachricht unverändert erhalten wurde.

MAC-then-Encrypt (2)

- Ein Vorteil von MAC-then-Encrypt gegenüber Encrypt-and-MAC besteht darin, dass der MAC für Angreifer nicht sichtbar ist und somit auch keine Information preisgeben kann.
- Ein Nachteil von MAC-then-Encrypt ist die Notwendigkeit, eine gegebenenfalls manipulierte Nachricht zunächst entschlüsseln zu müssen.
- Theoretisch wäre es so möglich, dass Schadcode entschlüsselt wird und, wenngleich die MAC-Prüfung dann fehlschlägt, dieser Code beim lesenden Zugriff in einem System zur Ausführung kommt.
- Ein weiterer Nachteil ist, dass unter Umständen Chosen Ciphertext Attacken ermöglicht werden, falls ein Angreifer Zugriff auf die Entschlüsselungsfunktion besitzt.
- Mac-then-Encrypt kam im TLS Protokoll zum Einsatz. Ab Version TLS 1.3 arbeitet TLS stattdessen mit sogenannten Authenticated Ciphers.

Encrypt-then-MAC

- Bei ***Encrypt-then-MAC*** wird der Message Authentication Code auf die bereits verschlüsselte Nachricht angewandt:
  - Der Sender berechnet erst $C = E_K(P)$ und dann $MAC_{K'}(C)$.
  - Das Ergebnis C, $MAC_{K'}(C)$ wird an den Empfänger übermittelt.
  - Der Empfänger prüft den MAC-Wert. Ist dieser korrekt, entschlüsselt der Empfänger die Nachricht C und erhält $P = E^{-1}_K(C)$.

- Ein Vorteil von Encrypt-then-MAC besteht darin, dass manipulierte und ggf. schadhafte Inhalte bereits vor der Entschlüsselung erkannt und verworfen werden können.
- Die IPSec Protocol Suite macht Gebrauch von Encrypt-then-MAC.

MAC aufbewahren

- Im Einzelfall kann es gewünscht sein, dass eine Nachricht nicht nur nach Erhalt auf Unverfälschtheit überprüft werden kann, sondern auch später noch, nachdem sie bereits abgespeichert wurde.
- Die Variante Encrypt-then-MAC wäre für diesen Fall ungeeignet, es sei denn man möchte die verschlüsselte Nachricht aufbewahren. Anderenfalls geht der MAC verloren.
- Die Variante MAC-then-Encrypt hingegen unterstützt die Möglichkeit, entschlüsselte Daten inklusive zugehörigem MAC abzuspeichern.
- Ein Vergleich der unterschiedlichen Varianten muss also nicht nur theoretische Angriffspotentiale berücksichtigen, sondern auch die Anforderungen des jeweiligen Einsatzszenarios.

MAC ohne Verschlüsselung

- In manchen Anwendungen spielt die Vertraulichkeit nur eine untergeordnete Rolle, während die Integrität und Authentizität von hoher Bedeutung sind.
- Beispiel: in Industrial Control Systems muss sichergestellt werden, dass Steuerbefehle nicht manipuliert werden können. In der Praxis gibt es hier leider nur selten adäquate Schutzmechanismen.
- Beispiel: das Simple Network Management Protocol Version 3 bzw. SNMPv3. Dieses Protokoll übermittelt Konfigurationsanweisungen. Deren Schutz vor Manipulationen ist aus Sicherheitsgesichtspunkten unverzichtbar. Eine Verschlüsselung ist unabhängig davon möglich, aber nicht zwingend. Möchte man sich beispielsweise davor schützen, dass Angreifer mit Tools wie Wireshark den Netzwerkverkehr analysieren, so kann zusätzlich verschlüsselt werden. Die angebotenen Varianten sind:
  - No authentication and no privacy (noAuthNoPriv) – z.B. für Monitoring
  - Authentication and no privacy (authNoPriv) – z.B. für Kontrollbefehle
  - Authentication and privacy (authPriv) – z.B. für Download vertraulicher Daten

Authenticated Ciphers

- ***Authenticated Ciphers*** vereinen innerhalb eines Verfahrens die Funktionalitäten Verschlüsselung und Authentizität. Dies führt zu einer Vereinfachung und reduziert die Fehleranfälligkeit. Andererseits wird die Flexibilität reduziert, da nicht unterschiedliche Mechanismen für Encryption und für MAC miteinander kombiniert werden können.
- Man spricht hier auch von ***Authenticated Encryption*** bzw. ***AE***.
- Beispiele für Authenticated Ciphers werden wir später kennenlernen.

Authenticated Encryption with Associated Data

- Authenticated Encryption, ebenso wie die zuvor beschriebenen Varianten, sichern nicht nur die Integrität bzw. Authentizität einer Nachricht, sie verschlüsseln diese auch komplett. Je nach Anwendung kann es jedoch erforderlich sein, einen Teil der Daten unverschlüsselt zu lassen.
- Beispiel: soll ein Datenpaket für den Transport im Netzwerk verschlüsselt werden, so müssen die Empfängerangaben (und ggf. weitere Daten) unverschlüsselt bleiben, da ansonsten keine Zustellung möglich wäre. Dennoch soll die Gesamtheit aller Daten (einschließlich Empfängerangaben) mittels Message Authentication geschützt werden vor Manipulation.
- Solche unverschlüsselt bleibenden Nachrichtenbestandteile nennt man ***Associated Data***. Das hier beschriebene Vorgehen wird bezeichnet als ***Authenticated Encryption with Associated Data*** bzw. ***AEAD***.
- Bemerkung: Da viele Authenticated Encryption Verfahren de facto auch Associated Data erlauben, verwendet man nicht immer den gesonderten Begriff AEAD sondern spricht auch hier einfach nur von Authenticated Encryption bzw. AE.

Cryptographic Checksums

- Eine kryptographische Prüfsumme bzw. ein MAC muss, im Gegensatz zu einem Verschlüsselungsalgorithmus, nicht bijektiv sein.
- Wir kennen bereits ein Verfahren zur Bildung einer kryptographischen Prüfsumme: wir bilden den Hashwert einer Nachricht und wenden sodann eine digitale Signatur an auf diesen Hashwert.
- Kryptographische Prüfsummen müssen freilich nicht auf Basis von Public Key Verfahren konstruiert sein. Symmetrische Verfahren, bei denen Sender und Empfänger einen gemeinsamen geheimen Schlüssel teilen, sind ebenso möglich.
- Im symmetrischen Fall haben wir drei verschiedene Parameter für die zugehörigen Stringlängen in Bits:
  - Länge k des symmetrischen Schlüssels
  - Länge t des MAC
  - Länge m der Nachricht
- Wir betrachten im folgenden, welche Optionen dies für Brute Force Angriffe ermöglicht.

Brute Force Angriffe (1)

- Betrachten wir zunächst den Fall, dass wir eine Prüfsumme haben, aber weder die zugehörige Nachricht noch den für die Prüfsumme verwendeten Schlüssel kennen.
- Ohne in einem ersten Schritt die zugrunde liegende Nachricht zu finden, macht es wenig Sinn, den verwendeten Schlüssel zu suchen.
- Angenommen, unsere Prüfsumme hat 128 Bit Länge und die zugehörige Nachricht besteht aus nur zehn Blöcken zu je 128 Bit, hat insgesamt also eine Länge von 1280 Bit. Dann gibt es insgesamt 2^{1280} mögliche Inputs, die auf 2^{128} mögliche Outputs abgebildet werden.
- Statistisch werden also jeweils etwa $2^{1280}/2^{128} = 2^{1280 - 128} = 2^{1152}$ Nachrichten auf dieselbe Prüfsumme abgebildet.
- Aus dem Beispiel wird klar, dass es kaum möglich ist, zu gegebener Prüfsumme die “richtige” Nachricht zu finden, auch wenn Nachricht und Prüfsumme für irgendeinen Schlüssel zueinander passen sollten und die Nachricht einen korrekt formatierten, sinnvollen Inhalt darstellt.
- Es existieren schlicht zu viele solche Nachrichten, die diesen Anforderungen genügen.

Brute Force Angriffe (2)

- Nun betrachten wir den Fall, dass wir eine Nachricht M im Klartext nebst zugehöriger Prüfsumme T haben. Nun suchen wir den für die Prüfsumme verwendeten Schlüssel K.
- Angenommen, der Schlüssel ist länger als die kryptographische Prüfsumme: $k > t$. Ein Angreifer kann nun (theoretisch) alle Schlüssel K der Reihe nach ausprobieren und testen, ob $MAC_K(M) = T$ ergibt.
- Wir berechnen für gegebenes M auf diese Weise 2^k Werte $MAC_K(M)$. Diese können allerdings nicht alle verschieden sein, da es insgesamt nur 2^t unterschiedliche Prüfsummen der Länge t Bit gibt.
- Statistisch müssen also rund $2^k/2^t = 2^{k-t}$ verschiedene Schlüssel dazu führen, dass M auf T abgebildet wird. Im Fall $k > t$ genügt ein Paar (M, T) also nicht, um eindeutig den Schlüssel K zu ermitteln.
- Haben wir ein zweites Paar(M', T'), so reduziert sich die Zahl passender Schlüssel erneut, in analoger Weise: sollte k-t noch immer größer sein als t, so werden rund $2^{k-t}/2^t = 2^{k-2t}$ verschiedene Schlüssel dazu führen, dass M' auf T' abgebildet wird.
- Insgesamt benötigen wir daher i Paare aus Nachricht und Prüfsumme, so dass $k - it < 0$ und mit hoher Wahrscheinlichkeit nur noch genau ein Schlüssel alle diese Nachrichten auf die zugehörigen Prüfsummen abbildet.
- Falls $k < t$, so kann (muss jedoch nicht) schon ein Paar (M, T) zur Identifizierung des korrekten Schlüssels ausreichen.

Brute Force Angriffe (3)

- In der Praxis sind Brute Force Angriffe bei hinreichend großer Schlüssellänge nicht möglich. Wir fassen zusammen:
- Brute Force Attacken auf Message Authentication Codes haben zum Ziel, für selbst gewähltes M einen gültigen Wert $MAC_K(M)$ zu erhalten.
- Für den Angriff müssen ein oder mehrere Paare aus Nachricht und zugehöriger Prüfsumme vorliegen, um auf Übereinstimmung testen zu können.
- Um den korrekten MAC Key zu identifizieren ist es notwendig, aber nicht immer hinreichend, alle Schlüssel K an einem vorliegenden Beispiel aus Nachricht M und Prüfsumme T auszuprobieren, bis $MAC_K(M) = T$ gilt. Da auch für weitere Schlüssel K' gelten kann, dass $MAC_{K'}(M) = T$, müssen im Bedarfsfall mehrere Paare aus Nachricht und korrekter zugehöriger Prüfsumme zum Testen vorliegen, um zu entscheiden, welcher der Keys der richtige ist.
- Ist der MAC kürzer als der geheime Key, so verringert das nicht automatisch den Aufwand, da ggf. viele Keys den korrekten MAC zu einer vorgegebenen Nachricht liefern.
- Nicht alle erfolgreichen Angriffsmethoden müssen den MAC Key berechnen.