Kellinghusen, 21.04.2022

Datensicherheit in der „Cloud“ – Fragen an den Sachverständigen Andreas Bethke

Um unsere Sichtweise auf die Definition des Begriffs „Ende-zu-Ende-Verschlüsselung“ (E2EE) und deren Wirksamkeit im Hinblick auf Datensicherheit überprüfen zu lassen, haben wir dem mehrfach akkredierten Sachverständigen Andreas Bethke eine Reihe von Fragen vorgelegt und um eine Stellungnahme gebeten.

Kann eine serverseitige SaaS (Webanwendung) E2EE sein? Welche Rolle kann „Key-management“ in diesem Zusammenhang spielen?

Um die Frage im richtigen Kontext beantworten zu können, muss zunächst der Begriff „E2EE“ (End-to-End-Encryption / Ende-zu-Ende-Verschlüsselung) erläutert werden. Grundsätzlich wird im Rahmen von Verschlüsselungsverfahren ganz allgemein zwischen einer „Transportverschlüsselung“ und einer „Inhaltsverschlüsselung“ unterschieden.

Bei der Transportverschlüsselung werden unverschlüsselte Daten über einen verschlüsselten Kanal (gerne auch als Tunnel bezeichnet) gesendet. Vor dem Senden der Daten wird dabei ein gezielter Tunnel (zwischen zwei Systemen, Geräten oder zwischen Webserver und Browser) aufgebaut und die Daten werden für den Transport verschlüsselt. Bei Webanwendungen kommt hierbei das Transport Layer Security (TLS)-Protokoll zum Einsatz. Dadurch wird verhindert, dass Unbefugte auf der Transportstrecke Daten auf einfache Art und Weise mitlesen oder verändern können.

Die gesonderte Ver- oder Entschlüsselung der Daten innerhalb einer Anwendung ist dabei nicht vorgesehen.

Der bzw. die benötigten Schlüssel für solche Verfahren werden automatisch zwischen einem Webserver und einem Webbrowser mit Hilfe von Zertifikaten ausgetauscht.

Im Gegensatz dazu steht die Inhaltsverschlüsselung, bei der zunächst die Daten mit verschlüsselt werden. Der Schlüssel hierfür befindet sich im Besitz desjenigen, der die Verschlüsselung angestoßen hat. Im nächsten Schritt werden die so verschlüsselten Daten gespeichert oder übermittelt. Bei der Übermittlung wird häufig noch eine zusätzliche Transportverschlüsselung genutzt. Eine Entschlüsselung ist nur demjenigen möglich, der genau den Schlüssel besitzt, mit dem die Daten verschlüsselt werden.

Auf den ersten Blick scheinen diese beiden Methoden keine großen Unterschiede zu haben. Vergleicht man jedoch diese Verfahren am Beispiel einer E-Mail-Verschlüsselung, wird der Unterschied klar: Beim Verschicken von E-Mails von einem Sender an einen Empfänger bauen sowohl die Mail-Clients zum Mailserver, als auch die Mailserver untereinander immer TLS-Tunnel auf. Da die Server jedoch nicht direkt verbunden sind, sondern sich innerhalb eines (weltweiten) Netzwerkes befinden, wird bei dieser Methode an jedem einzelnen Knotenpunkt des Netzwerkes (also den beteiligten Servern) stets ein neuer Tunnel aufgebaut. Die E-Mail ist dabei auf jedem dieser beteiligten Server unverschlüsselt und kann

dort mitgelesen werden, so wie auch nach Ankunft der E-Mail auf dem Zielserver. So kann der Empfänger den Inhalt der E-Mail sofort lesen.

Wird jedoch die E-Mail vor dem Senden ausschließlich für den Empfänger verschlüsselt, so bleibt deren Inhalt allen Stationen auf dem Weg vom Sender zum Empfänger verborgen. Erst im Empfänger-Mail-Client wird dann die E-Mail entschlüsselt und wieder sichtbar gemacht.

Dies nennt man eine echte Ende-zu-Ende-Verschlüsselung (Inhaltsverschlüsselung). Sie setzt allerdings voraus, dass es immer einen Sender und einen (oder mehrere) Empfänger gibt. Oft wird hierbei ein Schlüsselpaar benutzt, das aus einem privaten und einem öffentlichen Teil besteht (sog. „Public-Key-Verfahren"). Der öffentliche Schlüsselteil (des bzw. der Empfänger) ist dem Sender bekannt, der private Schlüsselteil hingegen nur dem einzelnen Empfänger selbst. Eine Entschlüsselung ist somit nur mit dem privaten Schlüsselteil möglich.

Bezieht man dieses Konstrukt auf die Nutzung eines Cloud-Speichers (wie z.B. Dropbox, Google Drive oder OneDrive, deren Verbindungen immer TLS-gesichert sind), müssten die Daten vor dem Abschicken verschlüsselt werden und könnten erst nach dem Herunterladen aus dem Cloudspeicher entschlüsselt werden. Wenn man so verführe, könnte man auch hier von einer echten Ende-zu-Ende-Verschlüsselung reden, auch wenn der Empfänger (derjenige, der die Daten aus der Cloud lädt und entschlüsselt) zum Zeitpunkt des Verschlüsselns und Sendens ggf. noch gar nicht feststeht.

Sofern eine Transportverschlüsselung zum Einsatz kommt, wird auch von einer Verschlüsselung „in-transit" gesprochen („Data-in-transit").

Aus diesen Ausführungen wird klar, dass eine Webanwendung, die auf einem Server ausgeführt wird, d.h. ihre Daten im Wesentlichen auf dem Server selbst verarbeitet und dem aufrufenden Webbrowser die Verarbeitungsergebnisse (und damit die Daten) als Ansicht übermittelt, im Grundsatz keine Ende-zu-Ende-Verschlüsselung sein kann, da die Daten eben nicht inhaltsverschlüsselt gegenüber einem Empfänger sind, sondern (nur) transportverschlüsselt. Diese Aussage bezieht sich ausschließlich auf und für im Webbrowser dargestellten Inhalte.

So mag es innerhalb der Webanwendung durchaus Ende-zu-Ende-Verschlüsselungsmechanismen geben, um z.B. Nachrichten an den Benutzer zu verschlüsseln, die nur er mit einem ihm bekannten Passwort (oder sonstigen Schlüssel) entschlüsseln kann. Wichtigstes Element ist dabei das angesprochene „Schlüsselmanagement". Es ist daher zwingend notwendig, dass der Benutzer (in seiner Funktion als Empfänger), und auch nur er, den Schlüssel zum Entschlüsseln kennt oder besitzt.

Zudem kann es innerhalb der Webanwendung die Möglichkeit geben, dass die Daten außerhalb des Webservers lagern (z.B. auf einem entfernten Datenbankserver) und dass dort verschlüsselt abgelegt sind.

Dies stellt jedoch keine E2EE dar, da eine Ende-zu-Ende-Verschlüsselung immer nur im Kontext eines „Sender-Empfänger-Modells" zum Tragen kommt (bspw. nach Shannon und Weaver). In diesem Modell ist ein Empfänger (oder auch Adressat) eine Person, oder eine Institution, die eine Nachricht oder eine andere Information durch ein Medium eines

Absenders übermittelt bekommt. Als Empfänger im Kontext eines Sender-Empfänger-Modells kann jedoch in keinem Fall ein technisches Gerät (wie z.B. ein Webserver, E-Mail-Server, Datenbankserver oder ein Browser) verstanden werden.

Wie ist die folgende Aussage zu bewerten aus technischer Sicht mit Blick auf die Datensicherheit: „Wir sind selbstverständlich Ende zu Ende Verschlüsselt, das liegt in der Natur einer HTTPs-Verbindung. Der private Schlüssel für die Verschlüsselung der Daten ist bei uns im KMS gespeichert, was mMn besser ist, als unverschlüsselt auf dem System der Therapeuten. Das ist eine Architekturentscheidung. Bei einer Datenpanne mit Sicherheit die bessere Entscheidung."

Eine HTTPs-Verbindung ist eine abgesicherte Verbindung des sog. „Hypertext Transfer Protocols", also ein Internetkommunikationsprotokoll, das die Integrität und Vertraulichkeit des Datenverkehrs zwischen dem Webbrowsers eines Nutzers und dem Webserver schützen soll. Es handelt sich dabei um die in Frage 1 bereits erwähnte Transportverschlüsselung, die keine Ende-zu-Ende-Verschlüsselung im Kontext des Sender-Empfänger-Modells darstellt.

Ob das Produkt, auf das sich die Aussage bezieht, dennoch eine Ende-zu-Ende-Verschlüsselung nutzt, kann auf Grund der Aussage nicht beurteilt werden. Lediglich die Begründung ist falsch, dass jede durch TLS gesicherte Verbindung automatisch eine Ende-zu-Ende-Verschlüsselung beinhaltet.

Ferner ist die Rede von einem privaten Schlüssel, der für eine Verschlüsselung eingesetzt wird. Aus technischer Sicht ergibt diese Aussage keinen Sinn, denn im oben genannten „Private-Key-Verfahren", wird die Verschlüsselung immer mit einem öffentlichen (und somit allen Beteiligten bekannten) Schlüssel durchgeführt. Der private Schlüssel wird ausschließlich zur Entschlüsselung genutzt. Eine externe Speicherung eines solchen Schlüssels beispielsweise in einem „Key-Management- System" (KMS) kann sinnvoll sein, aber nur, wenn der Besitzer des privaten Schlüssels hierüber die Kontrolle hat. Laut der Aussage ist dies jedoch nicht der Fall.

Der letzte Teil der Aussage hinsichtlich der Auswirkung einer Datenpanne in diesem Zusammenhang kann nicht bewertet werden, da die Datenpanne (gemeint ist vermutlich eine Verletzung des Schutzes von personenbezogenen Daten gem. DSGVO) nicht näher beschrieben ist. Dies würde zu vielen Spekulationen und Möglichkeiten solcher Datenpannen führen.

Ist der Einsatz einer AWS KMS-Architektur als gleichwertig in Bezug auf die Sicherheit anzusehen zu einer E2EE, wenn der Auftragsverarbeiter Zugriff auf die im KMS hinterlegten Schlüssel hat?

Kurze Antwort: Nein.

Zunächst muss erläutert werden, was es mit der KMS-Architektur der Amazon Web Services (kurz AWS) auf sich hat und welche Schlüssel dort verwaltet werden.

Es handelt sich bei dem AWS KMS um einen zentralen Dienst, um kryptographische Schlüssel (symmetrische und asymmetrische) zu erzeugen und zu verwalten. Dabei gewährleistet AWS, dass es auch Mitarbeitern von AWS nicht möglich ist die erzeugten und genutzten Schlüssel von AWS-Kunden im Klartext zu exportieren oder anzeigen zu lassen.

Zur Erläuterung: Symmetrische Schlüssel kommen zum Einsatz, wenn Daten mit dem gleichen Schlüssel ver- und wieder entschlüsselt werden. Asymmetrische Schlüssel kommen bei der bereits erwähnten echten Ende-zu-Ende-Verschlüsselung zum Einsatz, wie beispielsweise bei der Verschlüsselung von E-Mails mittels PGP (Pretty Good Privacy) oder GPG (GNU Privacy Guard).

In der Regel gibt es drei Szenarien zur Datenverschlüsselung mit dem AWS KMS: Erstens können Kunden von AWS das KMS durch sog. Application Programming Interfaces (APIs) direkt zur Ver- und Entschlüsselung von Daten mithilfe ihrer im Service gespeicherten KMS-Schlüssel verwenden.

Zweitens können die AWS-Kunden ihre Daten von AWS-Services mithilfe der im Service gespeicherten KMS-Schlüssel verschlüsseln lassen. In diesem Fall werden Daten mit Datenschlüsseln verschlüsselt, die durch Ihre KMS-Schlüssel geschützt werden.

Drittens können die AWS-Kunden das AWS Encryption SDK (Software Development Kit) verwenden, das in AWS KMS integriert ist, um die Verschlüsselung innerhalb Ihrer eigenen Anwendungen durchzuführen, egal, ob diese auf AWS ausgeführt werden oder nicht.

Sofern es um die Schlüssel innerhalb des AWS KMS geht, sind diese dort sicher verwahrt, unabhängig davon, welches Szenario gewählt wird. Technisch werden hierfür sogenannte zertifizierte „Hardware Security Moduls“ (HSM) genutzt, die ein Auslesen durch Unbefugte wirksam verhindern.

Zudem verhindert KMS, dass symmetrische Schlüssel im Klartext exportiert werden können. Allerdings können die symmetrischen Datenschlüssel mithilfe der "GenerateDataKey"-API oder der "GenerateDataKeyWithoutPlaintext"-API exportiert werden. Außerdem können der private und der öffentliche Teil asymmetrischer Datenschlüsselpaare aus AWS KMS mithilfe der "GenerateDataKeyPair"-API oder der "GenerateDataKeypairWithoutPlaintext"-API exportiert werden.

Fazit: Das KMS von AWS ist zunächst als sicher zu betrachten, wenn dort Schlüssel (zum Entschlüsseln) hinterlegt sind. Vor dem Hintergrund, dass Daten nur vom Besitzer für sich selbst verschlüsselt werden sollen (Sender = Empfänger), ist der Einsatz eines KMS kein Sicherheitsgewinn, wenn dem Auftragsverarbeiter der Zugriff auf diesen Schlüssel erlaubt ist und er somit ebenfalls jederzeit die Daten entschlüsseln kann.

Warum wird zwischen „Transportverschlüsselung“ und „echter E2EE“ unterschieden? Was sind die Merkmale einer „echten E2EE“?

Wie bereits in der Antwort von Frage 1 erläutert kommt es auf den Zweck der Verschlüsselung an.

Die Kernfrage ist dabei: Was ist das Ziel der Verschlüsselung? D.h. vor wem möchte ich unbefugten Zugriff schützen?

Bei der Transportverschlüsselung werden die Daten auf dem Weg zwischen zwei Systemen vor dem Verändern und der Kenntnisnahme Dritter geschützt. Dabei passiert der Austausch von Schlüsseln automatisch und unter Einsatz von Zertifikaten (meist ein sog. TLS-Zertifikat, das von einer vertrauenswürdigen Ausgabestelle verwaltet wird). Die Daten selbst sind vor und nach der Übermittlung (also dem Transport) unverschlüsselt, sofern sie nicht durch weitere Verschlüsselungsmechanismen verschlüsselt wurden.

Das bedeutet, dass die Daten auf dem Zielsystem sofort nach Erhalt automatisch im Klartext vorliegen. Eine weitere Schlüsseleingabe des Empfängers ist nicht notwendig.

Bei der echten Ende-zu-Ende-Verschlüsselung sind die Daten vor jeglichem unbefugten Zugriff geschützt, bis sie durch den Empfänger autorisiert mit dem richtigen Schlüssel entschlüsselt werden. Er muss zwingend über den (privaten) Schlüssel verfügen.

Die Merkmale einer echten E2EE sind wie unter Frage 1 aufgeführt:

- Die Form einer Inhaltsverschlüsselung (nicht Transportverschlüsselung), also z.B. für Austausch von verschlüsselten Daten über gesicherte oder ungesicherte Kanäle.
- Das Vorhandensein eines Senders und eines oder mehrerer Empfänger im Sinne des Sender-Empfänger-Modells (keine Systeme unter oder miteinander).
- Das Vorhandensein eines asymmetrischen Schlüssels, bei dem der Sender nur den öffentlichen Schlüssel des oder der Empfänger kennt, aber nicht die privaten Schlüssel. Ausnahme: Es sei denn, dass der Sender auch gleichzeitig einer der Empfänger ist.

Verringert oder steigert eine E2EE-verschlüsselte local-first SaaS die Risiken für die Anwender aus technischer Sicht?

Um die Frage beantworten zu können, soll zunächst beschrieben werden, was unter einer „local-first“ Software zu verstehen ist.

Eine „local-first“ Anwendung ist eine besondere Form einer Cloud-Anwendung. Bei typischen serverseitigen Cloud-Anwendungen werden die Daten auf dem Server als der primäre, maßgebliche Bestand der Daten behandelt. Wenn ein Client eine Kopie der Daten hat, handelt es sich lediglich um einen Cache, der dem Server untergeordnet ist. Jede Datenänderung muss an den Server gesendet werden, sonst gilt sie als „nicht passiert“. Bei

clientseitigen „local-first-Anwendungen“ werden diese Rollen getauscht: Die Kopie der Daten auf dem lokalen Gerät - Laptop, Tablet oder Telefon – wird als der primäre Bestand behandelt. Server gibt es zwar immer noch, aber sie enthalten sekundäre Kopien der Daten, um den Zugriff von mehreren Geräten aus zu unterstützen.

Wie bereits im Vorwege ausgeführt, kann bei einer echten Ende-zu-Ende-Verschlüsselung der Sender auch der Empfänger sein. Im Falle einer „local-first“-Anwendung kommt dem eine besondere Bedeutung zu, wenn davon ausgegangen wird, dass die Daten(-Kopie) dauerhaft für den Empfänger verschlüsselt ist und somit auch die Kopie, die auf dem Server liegt. Sofern sich der Benutzer auf einem anderen Gerät anmeldet, wird die Kopie des Servers auf das neue Gerät übermittelt und erst wenn der Benutzer seinen (privaten) Schlüssel eingegeben hat, liegen die Daten wieder (temporär) im Klartext vor.

Zu den technischen Risiken:

Sofern der Benutzer kein eigenes (ausgelagertes) KMS, wie etwa einen Passwort- oder Schlüsselsafe) verwendet, kann es sein, dass er im Falle des Passwort-Verlustes (durch Vergessen) den Zugriff auf seine Daten verliert, sofern keine Art von „Notfall- Schlüssel“ vorgehalten wird. Somit erhöht dieser Aspekt das technische Risiko.

Auf der anderen Seite wird das Risiko von unbefugter Kenntnisnahme durch eine verschlüsselte lokale Datenhaltung stark vermindert. Selbst der Verlust des Gerätes führt nicht zwangsläufig zu einem Verlust der Daten, da eine Kopie auf dem Server gespeichert bleibt.

Diese beiden Risiken gilt es bei der Betrachtung und die Waagschale zu werfen und durch flankierende Maßnahmen wie oben beschrieben zu minimieren.

Gibt es zwischen verschiedenen Architekturen (nativ, serverseitige SaaS, clientseitige SaaS) Unterschiede in den Risiken und der Erfolgswahrscheinlichkeit hinsichtlich digitaler oder analoger Angriffe auf die Anwendergeräte?

In den letzten Jahren haben die digitalen Angriffe massiv zugenommen. Analoge (oder besser gesagt physische) Angriffe wie Diebstahl oder Zerstörung von Geräten kommen zwar auch vor, jedoch rückt der Fokus dabei eher auf kleine Geräte wie Smartphones oder Tablets.

Dabei gibt es folgende Risiken:

Bei physischen Angriffen könnten Unbefugte Zugriff auf Daten bekommen. Insbesondere Smartphones und Tablets können über einen relativ kurzen Zahlencode oder eine Muster-eingabe freigeschaltet werden. Hinzu kommen in den Webbrowsern (oder auf den Geräten) gespeicherte Zugangsdaten zu Plattformen, sofern der Benutzer dies aus Bequemlichkeits-gründen eingerichtet hat. Voraussetzung ist in diesen Fällen jedoch immer der physische Zugriff auf die Geräte.

Ein Zugriff auf Daten, die serverseitig gespeichert sind, wäre auf diesem Weg leicht möglich

und die Erfolgswahrscheinlichkeit entsprechend hoch, sofern der Zugriff nicht durch einen zweiten Faktor auf einem getrennten Kanal zusätzlich gesichert ist.

Wenn die Daten verschlüsselt auf dem Client liegen gilt das Gleiche, sofern die Möglichkeit zum Speichern des Schlüssels nicht unterbunden wurde.

Bei den digitalen Angriffen, die in der Regel über eine E-Mail erfolgen, wird versucht sog. Ransomware oder Keylogger auf das Gerät zu bringen.

Im Fall von Ransomware wird dabei das Gerät verschlüsselt. Für die Betrachtung der Fragestellung hat dies nur bei einer nativen Architektur einen Einfluss, wenn die Daten ausschließlich lokal auf dem Gerät gespeichert werden.

Keylogger oder ähnliche Trojaner hingegen sind gefährlicher, denn hier besteht das Risiko, dass Angreifer Daten unbefugt mitlesen können.

Dabei ist der Architekturansatz egal, da die Angreifer oft das sehen, was auch der Benutzer sehen kann. Ggf. ist das Risiko bei serverbasierten Lösungen noch größer, da die Angreifer nur die Benutzereingaben abfangen müssten, um Zugang zu der SaaS zu bekommen. Es sei denn, dass diese durch diesen (oben bereits genannten) weiteren Faktor, wie eine SMS-TAN abgesichert wäre. Angriffe per E-Mail haben derzeit die größten Chancen auf Erfolg.

Ist es von der Sache her gerechtfertigt, bei einer local bzw. offline first Web App von „weiteren Vorteilen" gegenüber einer serverseitigen SaaS zu sprechen? Unabhängig von der Frage, ob es unter UWG-Gesichtspunkten unlauter sein könnte, dass als Anbieter zu bewerben.

Das kommt auf den Kontext der Nutzung und eventuell auch äußere Parameter an. Verfügt der Anwender nur über eine eingeschränkte oder langsame Internetverbindung, so ist die Nutzung einer „local"-Web App ein klarer Vorteil. Gleiches gilt für den Einsatz möglicher Verschlüsselungsmechanismen, sofern dies notwendig ist.

Es wird mit Sicherheit weitere Vorteile geben, wenn es insbesondere um die Übernahme der Verantwortung über die Daten geht. Diese müssten aber im Detail (z.B. anhand von Funktionen oder Features) aufgelistet und bewertet werden. Pauschal lässt sich hier keine objektive Bewertung durchführen.

Inwieweit lässt sich aus den Bestimmungen der DSGVO ableiten, dass eine E2EE die technisch höherwertige Lösung gegenüber Transportverschlüsselung plus Data-at-rest-Verschlüsselung darstellt, um das Risiko einer Beeinträchtigung der Rechte und Freiheiten von Betroffenen (hier: Patienten von Heilberuflern) zu minimieren? Unter welchen Voraussetzungen kann das der Fall sein? (Greift die Antwort auf Frage 5 auf.)

Zu Klärung der Begrifflichkeit soll vorab die „Data-at-rest"-Verschlüsselung beschrieben werden.

Neben den unter Frage 1 beschriebenen Möglichkeiten zur Datenverschlüsselung, gibt es noch die Möglichkeit Daten zu verschlüsseln, die sich im Ruhezustand befinden. Dann spricht man von einer sog. Verschlüsselung „at-rest“. Dabei bezieht sich diese Art von Verschlüsselung meist auf ganze Festplatten oder SAN (Storage Area Network). Ein gutes Beispiel ist die „at-rest“ Verschlüsselung von Windows mit dem integrierten Bitlocker. Sofern das Windowssystem nicht läuft, ist die Festplatte verschlüsselt und mit ihr alle Daten, die darauf gespeichert sind. Nach dem Start von Windows kann auf die Daten der Festplatte zugegriffen werden. Dateien können ohne die Eingabe von Schlüsseln genutzt werden oder kopiert werden. In Server-Anwendungen werden Festplatten(bereiche) verschlüsselt, damit die Daten vor unbefugten Zugriffen geschützt sind, wenn die Server heruntergefahren sind.

Die Artikel 25, 32 und 35 der DSGVO führen an, dass unter dem Aspekt des risikobasierten Ansatzes Maßnahmen ergriffen werden müssen, um ein mögliches Risiko zu Beeinträchtigung der Rechte und Freiheiten von Betroffenen zu minimieren. Weiterhin führen die Artikel in diesem Kontext aus, dass der Einsatz einer Verschlüsselung eine adäquate Maßnahme ist. Dabei meint Verschlüsselung hier eine Inhaltsverschlüsselung (bei der der Verantwortliche im Sinne der DSGVO der Herr über den Schlüssel ist) und nicht nur eine Transportverschlüsslung.

Sofern es sich um Patientendaten, Gesundheitsdaten oder besser gesagt um besondere Kategorien personenbezogener Daten gem. Art. 9 DSGVO geht, ist das Risiko per se als hoch einzustufen, so dass risikominimierende Maßnahmen ergriffen werden müssen. D.h. der Einsatz einer Inhaltsverschlüsselung wie oben beschrieben ist aus meiner Sicht die angezeigte Lösung, auch wenn dies derzeit gesetzlich nicht ausdrücklich vorgeschrieben ist.

Zur Person:

Andreas Bethke ist Diplom Informatiker (FH) für Softwaretechnik und hat seinen beruflichen Schwerpunkt seit 1996 im Bereich Datenschutz. Im Zeitraum 2002 bis 2018 war er akkreditierter technischer Sachverständiger für das Datenschutz- Gütesiegel des Unabhängigen Landeszentrum für Datenschutz Schleswig-Holstein (ULD). Seit 2008 ist er als „Certified European Privacy Expert Technical“ für das European Privacy Seal (EuroPriSe), sowie als technischer Gutachter für das ePrivacy-Seal tätig und hat in den vergangenen 20 Jahren an über 100 Datenschutz-Gutachten mitgewirkt. Er ist ein Kooperationspartner des deutschen „Datenschutz-Guru“ RA Stephan Hansen-Oest.

Darüber hinaus ist Andreas Bethke im Bereich der Informationssicherheit tätig und hier als Informationssicherheitsbeauftragter für BSI-Grundschutz und ISO 27001 zertifiziert und zum Lead-Auditor für ISO/IEC 27001 Zertifizierungen berufen.