Reden ist Silber – Schweigen ist Gold: Datensparsamkeit durch effizientes Rechnen unter Verschlüsselung

Thomas Schneider[1]

Kurzfassung:
Wo immer sensible Daten gespeichert werden zeigt die Praxis, dass es lediglich eine Frage der Zeit ist, bis diese durch ein Sicherheitsleck an unerwünschte Personen gelangen. Die einzige Möglichkeit, um dies zuverlässig zu vermeiden ist Datensparsamkeit, die Daten also erst gar nicht zu speichern. In vielen Fällen werden die Daten jedoch trotzdem für Berechnungen benötigt.
Dieser Beitrag gibt einen Überblick über Methoden des effizienten Rechnens unter Verschlüsselung und zeigt, wie diese ermöglichen, beliebige Berechnungen auf sensiblen, verschlüsselten Daten durchzuführen, ohne dass diese im Klartext gespeichert oder zu irgendeinem Zeitpunkt im Klartext vorliegen müssen. Insbesondere wird hierbei auf Erweiterungen durch sichere Hardware auch im Kontext des Rechnens in der „Cloud“, sowie die Unterstützung durch Werkzeuge eingegangen. Als mögliche Anwendungen werden Systeme zur Gesichtserkennung und Klassifikation medizinischer Daten betrachtet, welche die Privatsphäre erhalten.

Stichworte: Schutz der Privatsphäre, Rechnen auf verschlüsselten, Rechnen mit verschlüsselten Funktionen, Privacy Enhancing Technologies, Sicherheit und Datenschutz für Internetdienste, Nutzbarkeit von Sicherheitstechniken, Sicherheit in der Cloud, Embedded Security / Security by Design.

1. Einleitung

Fast täglich wird über Datenlecks berichtet, durch die sensible persönliche Daten an unerlaubte Benutzer oder gar die Öffentlichkeit gelangen [Zeit10] - die Dunkelziffer liegt wohl noch deutlich höher. Zwar sind Unternehmungen mittlerweile gesetzlich dazu verpflichtet, solche Vorkommnisse – falls sie entdeckt werden – öffentlich zu machen, doch lassen sich die hierdurch entstandenen Auswirkungen und Schäden weder rückgängig machen, noch beziffern. Die wohl zuverlässigste Möglichkeit, solche Datenlecks zu vermeiden ist Datensparsamkeit, also sensible Daten erst gar nicht zu speichern. Oft ist dies jedoch nicht möglich, da die Daten für Berechnungen benötigt werden.

Ein nahe liegender Ansatz ist es, die Daten (symmetrisch oder asymmetrisch) verschlüsselt zu speichern, doch müssen diese zur Verarbeitung dann zunächst entschlüsselt werden. Zu diesem Zeitpunkt haben Insider (z.B. Systemadministratoren) jedoch Zugriff auf die entschlüsselten Klartextdaten, so dass Datenlecks somit nicht vollständig vermieden werden können.

Rechnen unter Verschlüsselung erlaubt es, Berechnungen auf verschlüsselten Daten durchführen zu können, ohne die Daten zwischenzeitlich zu entschlüsseln.

In diesem Beitrag werden die im Rahmen meiner Diplomarbeit [1-3] entwickelten Methoden zum effizienten Rechnen unter Verschlüsselung zusammengefasst und

[1] Horst Görtz Institut für IT-Sicherheit, Ruhr-Universität Bochum

gezeigt, wie diese für vielfältige Szenarien und Anwendungen weiter verbessert und praktisch nutzbar gemacht werden können, dem Forschungsgegenstand meiner Promotion [4-13].

2. Effizientes Rechnen unter Verschlüsselung

Das *Rechnen unter Verschlüsselung* erlaubt es zwei Parteien, Client C und Server S, eine beliebige, beiden bekannte Funktion f auf ihren geheimen Eingaben x bzw. y so zu berechnen, dass lediglich das Ergebnis f(x,y) bekannt wird, jedoch keine zusätzliche Information über die Eingabe der anderen Partei oder Zwischenergebnisse.

Hierfür gibt es zweierlei Ansätze (ausführlichere Beschreibung in [11]), verwendet im jeweiligen Werkzeug zur automatischen Generierung solcher Protokolle:

Beim "*Rechnen mit verschlüsselten Funktionen*" [Yao86] wird zusätzlich zu den verschlüsselten Daten, für jede der zu berechnenden Elementaroperationen eine verschlüsselte Übersetzungstabelle verwendet, die es erlaubt, eine beliebige Funktion einmalig auf den verschlüsselten Daten zu berechnen. Dieses Verfahren wird in Fairplay [MNPS04] verwendet.

Beim "*Rechnen auf verschlüsselten Daten*" [Pai99] werden die Daten mit einem speziellen Verschlüsselungsverfahren, sogenannter "homomorpher Verschlüsselung", verschlüsselt, welches es erlaubt, ohne Verwendung von Übersetzungstabellen, bestimmte Operationen auf den verschlüsselten Daten mehrmals zu berechnen. Dieses Verfahren wird in VIFF [DGKN09] benutzt.

Im Rahmen von [1] wurde das Verfahren von [Yao86] dahingehend verbessert, dass XOR Gatter nun sehr günstig ("*free XOR*") unter Verschlüsselung ausgewertet werden können [3]: die Auswertung eines XOR Gatters benötigt keine verschlüsselte Übersetzungstabelle mehr und der Berechnungsaufwand ist vernachlässigbar. Dies führt zu erheblichen Effizienzsteigerungen für zahlreiche Basisfunktionalitäten [8] wie Addition (Verbesserung um Faktor 4), Multiplikation (Faktor 2,5) sowie Vergleich (Faktor 2). Diese Verbesserungen erhielten Einzug in aktuelle Implementierungen zum Rechnen mit verschlüsselten Funktionen in Software [4,9] und Hardware [5,6].

Des Weiteren wurde in [1] gezeigt, wie zusätzlich die ausgewertete Funktion selbst geheim gehalten werden kann. Hierfür wird ein *universeller Schaltkreis* (Äquivalent zu einer universellen Turingmaschine) sicher ausgewertet, der jede beliebige Funktion simulieren kann, die als Boole'scher Schaltkreis mit k Gattern repräsentiert werden kann. Die existierende Konstruktion von Valiant [Val76] ist asymptotisch optimal mit einer Größe von O(k log k), jedoch sind die konstanten Faktoren zu groß für handhabbare Funktionsgrößen. Wir haben einen neuen universellen Schaltkreis entwickelt, der – abzüglich "free XORs" - lediglich $\Theta(1.5\ k \log^2 k)$ Gatter benötigt und somit in der Praxis verwendet werden kann. Die Praktikabilität dieser Konstruktion wurde durch eine Erweiterung des Werkzeugs Fairplay um private Funktionen, genannt FairplayPF, demonstriert [2].

3. Ergänzung durch (sichere) Hardware

Im Rahmen der Promotion wurden verschiedene Anwendungsszenarien untersucht, in denen Hardware verwendet werden kann, um die Effizienz von Protokollen zum Rechnen mit verschlüsselten Funktionen weiter zu verbessern.

3.1 Sichere Hardware-Token

In vielen Anwendungszenarien erstellt der Anbieter ein einbruchssicheres (tamper-proof) Hardware-Token, das er dem Kunden übergibt, der sich damit später authentisieren kann. Beispiele sind die SIM Karte im Mobiltelefon, Smartcards in Set-Top Boxen für Pay TV, oder von Banken ausgestellte EC Karten. Da diese Token sowohl einbruchssicher als auch kostengünstig sein sollen, sind ihre Ressourcen beschränkt (wenig Speicher und geringe Rechenleistung).

In [5] wurde gezeigt, wie ein solches Token verwendet werden kann, um die verschlüsselten Übersetzungstabellen zu generieren. Somit müssen diese nicht mehr über das Netzwerk übertragen werden und die Kommunikation ist unabhängig von der Größe der sicher berechneten Funktionalität. Das Token benötigt lediglich konstant viel Speicher und berechnet nur symmetrische kryptographische Operationen (AES, SHA-256).

3.2 Eingebettete Systeme

Wie in [6] gezeigt kann die Auswertung der verschlüsselten Funktion sehr effizient in Hardware implementiert werden. Die sichere Berechnung von AES auf einem FPGA (Altera Cyclone II, 50MHz, 8MB SDRAM) benötigte lediglich 144ms, während eine Implementierung in Software (Intel Core 2 Duo, 3GHz, 4GB RAM) 2s benötigt [4].

3.3 Rechnen in der „Cloud“

Das Rechnen in der „Cloud“ erlaubt das Anmieten von Speicher und Rechenleistung von externen Anbietern und verspricht eine Revolution der gesamten IT Landschaft. Eine offene Frage ist jedoch, ob und wie die „Cloud“ für sensible Daten genutzt werden kann. Es existieren bereits Ansätze, um sensible Daten in der Cloud sicher zu speichern und nach Schlüsselwörtern zu durchsuchen. Das sichere und praktikable Auslagern (Outsourcing) von Daten und *beliebigen* Berechnungen auf denselben (z.B. Statistiken oder komplexe Suchanfragen) ist jedoch noch ungelöst. Die Verfahren von [GGP10,CKV10] sind zwar asymptotisch effizient, jedoch heute noch nicht realisierbar, da die zu Grunde liegenden voll homomorphen Verschlüsselungsverfahren [Gen09a,Gen09b,DGHV10] noch nicht praxistauglich sind [SV10,GH10].

In [7] wurde gezeigt, wie ein sicheres Hardware-Token verwendet werden kann, um dies effizient zu lösen. Das Token (z.B. in Form des IBM Cryptographic Coprocessors 4764) wird in die Infrastruktur des Cloud-Anbieters integriert, jedoch lediglich in einer Vorberechnungsphase verwendet, während die Anfragen parallel in der Cloud berechnet werden.

4. Hybrides Rechnen unter Verschlüsselung

Die beiden in §2 zusammengefassten Verfahren zum Rechnen unter Verschlüsselung sind unterschiedlich gut für bestimmte Teil-Funktionalitäten geeignet. Das Rechnen mit verschlüsselten Funktionen benötigt lediglich eine konstante Anzahl an Nachrichten, ermöglicht die Komplexität in eine Vorberechnungsphase zu verschieben und erlaubt eine effizientere Berechnung von Vergleichsoperationen [8]. Beim Rechnen auf verschlüsselten Daten mit additiv homomorphen Verschlüsselungsverfahren wächst die Anzahl der Nachrichten zwar linear in der Anzahl der Multiplikationen, jedoch ist dieses Verfahren besser geeignet zur Multiplikation großer Zahlen [9].

In [8] wurde ein Framework vorgestellt, das die nahtlose Kombination beider Verfahren erlaubt, wodurch die jeweils effizienteste Methode für eine bestimmte Teil-Funktionalität gewählt werden kann. Dieses Framework wurde in einem Werkzeug umgesetzt, genannt TASTY (Tool for Automating Secure Two-partY computations) [9], das die benutzerfreundliche Beschreibung, automatische Generierung, sowie den automatischen Vergleich der Performanz solcher hybrider Protokolle ermöglicht. TASTY implementiert „free XORs“ (§2) und ermöglicht die automatische Vorberechnung, wodurch die Antwortzeit um Faktor 10 kürzer ist als in vorherigen Implementierungen [MNPS04], [4].

5. Mehr Datensparsamkeit in Anwendungen

Das Rechnen unter Verschlüsselung kann in vielfältigen Anwendungen verwendet werden, um sensible Daten während der Verarbeitung zu schützen. Exemplarisch wurden zwei Anwendungen untersucht, in denen sensible (biometrische bzw. medizinische) Eingabedaten von einem Server in verschlüsselter Form klassifiziert werden sollen. Für diese können mit Hilfe der in §4 beschriebenen hybriden Protokolle und dem zugehörigen Werkzeug TASTY effiziente Protokolle generiert werden.

5.1 Gesichtserkennung

In einem System zur Gesichtserkennung soll festgestellt werden, ob ein aufgenommenes Gesicht ähnlich einem anderen Gesicht in einer Datenbank von Gesichtern ist. Dies kann beispielsweise bei Grenzkontrollen oder biometrischen Zugangssystemen verwendet werden. Da die Datenbank der Gesichter sensible Daten enthält (z.B. Bilder von Verdächtigen) und meist in regelmäßigen Abständen geändert wird ist es nicht möglich, diese auf den Endgeräten zu speichern. Stattdessen wird in bisherigen Systemen das Anfragegesicht über einen sicheren Kanal an den Server übermittelt, der die Suche durchführt. Dies führt jedoch dazu, dass der Server das angefragte und eventuell ebenfalls sensible Gesicht im Klartext lernt. Mit einem System zur Privatsphäre erhaltenden Gesichtserkennung kann die Datenbank unter Verschlüsselung so durchsucht werden, dass sowohl die Datenbank als auch die Anfrage geheim bleiben. Hierfür wurde ein effizientes hybrides (s. §4) Protokoll entworfen [10] und mit TASTY automatisch generiert [9].

5.2 Klassifikation medizinischer Daten

Als medizinisches Anwendungsszenario wurde ein Web-Service betrachtet, der biometrische Daten unter Verschlüsselung klassifiziert. Hierfür ist die Klassifikation von Elektrokardiogramm (EKG) Daten ein repräsentatives Beispiel: Ein Patient misst zu Hause sein EKG Signal und möchte dies bei einem Serviceanbieter klassifizieren lassen. Hierbei soll jedoch gleichermaßen die Privatsphäre des Patienten (EKG als personenbezogene Patientendaten), sowie das geistige Eigentum des Serviceanbieters (Parametrisierung des Algorithmus zur Klassifikation von EKG Signalen) geschützt werden. Auch für diese Anwendung wurde ein effizientes hybrides Protokoll entworfen [11,12] und mit TASTY automatisch generiert [13].

6. Diskussion und Ausblick

Im Folgenden werden die Möglichkeiten, Grenzen und offenen Fragestellungen des Rechnens unter Verschlüsselung kurz diskutiert.

6.1. Effizienz

Protokolle zum Rechnen unter Verschlüsselung sind naturgemäß langsamer als wenn die Berechnungen unverschlüsselt durchgeführt werden. Als Faustregel für die Effizienz gilt, dass Protokolle zum Rechnen mit verschlüsselten Funktionen (s. §2) in der Online Phase in etwa eine symmetrische Entschlüsselung (z.B. Hashen eines Blocks mit SHA-256) pro Bit-Operation kosten, während Berechnungen unter additiv homomorpher Verschlüsselung (s. §2) langsamer als bestehende public-key Verfahren wie RSA sind.

Dank der rasanten Zunahme verfügbarer Rechenleistung und rapide steigender Übertragungsraten von Datennetzen, in Kombination mit mehreren algorithmischen Verbesserungen der zugrunde liegenden Primitiven und Protokolle (s. §2 und [4]), ist Rechnen unter Verschlüsselung jedoch in den Bereich der Anwendbarkeit in der Praxis gerückt (s. §5 und [OPJM10]).

6.2. Sicherheit

Der beim Rechnen unter Verschlüsselung verwendete Sicherheitsparameter bestimmt die Zeitspanne, bis zu der die Verschlüsselung als sicher angenommen werden kann. Die meisten existierenden prototypischen Implementierungen zum Rechnen unter Verschlüsselung basieren jedoch auf einem symmetrischen Sicherheitsparameter von 80 bit, dessen Verwendung nur noch bis Ende 2010 empfohlen wird [GQ10].

Insbesondere in Anwendungen, in denen sensible Daten (z.B. Patientendaten) unter Verschlüsselung verarbeitet werden, ist es notwendig, dass mitgeschnittene Protokollläufe auch in Zukunft nicht entschlüsselt werden können. Hierfür sollte ein ausreichend großer Sicherheitsparameter gewählt werden (z.B. 128 bit, dessen Sicherheit bis zum Jahr 2040 prognostiziert wird [GQ10]). Die Verwendung eines größeren Sicherheitsparameters hat jedoch insbesondere beim Rechnen auf verschlüsselten Daten erhebliche negative Auswirkungen auf die Performanz der Protokolle.

In manchen Anwendungen ist die in vielen Papieren getroffene Annahme, dass die beteiligten Parteien ehrlich aber neugierig (engl. “semi-honest adversaries”) sind, also nicht in bösartiger Absicht vom Protokoll abweichen, ebenfalls nicht gerechtfertigt. Bestehende Protokolle zum Rechnen unter Verschlüsselung, die solche Betrugsversuche mit hoher (engl. “covert adversaries”) oder an Sicherheit grenzender (engl. “malicious adversaries”) Wahrscheinlichkeit erkennen sind jedoch deutlich aufwändiger. Beispielsweise benötigt die sichere Berechnung der aus 33.880 Bit-Operationen bestehenden AES Funktionalität mit Sicherheitsparameter 128 bit im semi-honest Fall 7 s (0, 5 MByte), im covert Fall 60 s (8, 7 MByte) und im malicious Fall 19 min (408 MByte) [4].

6.3. Benutzbarkeit

Neben der Performanz und Sicherheit der Protokolle zum Rechnen unter Verschlüsselung ist ihre Benutzbarkeit der wohl wichtigste Aspekt für den Einsatz in praktischen Anwendungen, um Entwicklungskosten zu minimieren.

Für den Entwurf effizienter und sicherer Protokolle war lange Zeit ein detailliertes Expertenwissen nötig. Auch bei der Implementierung der Protokolle konnten sich diverse Programmierfehler einschleichen, welche die Sicherheit gefährdeten. Dies ist vergleichbar mit Zeiten, in denen ausschließlich in Maschinensprache (Assembler) programmiert wurde.

Compiler für kryptographische Protokolle (wie das in §4 beschriebene Werkzeug TASTY, Fairplay oder VIFF) abstrahieren von den kryptographischen Details und erlauben es so Anwendungsentwicklern auch ohne detailliertes kryptographisches Hintergrundwissen die zu berechnende Funktionalität in einer Hochsprache zu beschreiben. Aus dieser Hochsprachenbeschreibung wird dann automatisch ein ausführbares, effizientes und sicheres Protokoll erzeugt. In Analogie abstrahieren Hochsprachen wie Java, C oder C++ von maschinenspezifischen Details und generieren aus Programmen effizienten und korrekten (im Sinne von äquivalent zur Hochsprachenbeschreibung) Maschinencode und erhöhen somit die Produktivität.

Im EU Projekt CACE (Computer Aided Cryptography Engineering, http://www.cace-project.eu) wurden neben TASTY eine Vielzahl von Werkzeugen zur automatischen Generierung von kryptographischen Protokollen und Primitiven entwickelt, beispielsweise ein zertifizierender Compiler für Zero-Knowledge Beweise [13].

7. Zusammenfassung

Als Ausgangspunkt dieses Beitrags wurde betont, dass Datensparsamkeit oft unumgänglich ist, sensible Daten also am Besten nicht in unverschlüsselter Form gespeichert oder verarbeitet werden sollten (§1). Darüber hinaus können Daten mit Hilfe von modernen und effizienten Methoden des Rechnens unter Verschlüsselung sicher verarbeitet werden, ohne diese zu entschlüsseln (§2). Dies kann in verschiedensten Szenarien und auf unterschiedlichen Plattformen eingesetzt werden (§3). Die effizientesten, heute verfügbaren Protokolle zum Rechnen unter Verschlüsselung sind aus effizienten Grundbausteinen zusammengesetzt und können

mit Hilfe des Werkzeugs TASTY automatisch generiert werden (§4). Exemplarisch wurden zwei Anwendungen betrachtet, in denen mit Hilfe dieser Technologie die Privatsphäre geschützt werden kann (§5). Abschließend wurden die Möglichkeiten, Grenzen und offenen Fragen des Rechnens unter Verschlüsselung kurz diskutiert (§6).

Danksagung

Ich bedanke mich herzlich bei Prof. Dr. Volker Strehl und Dr. Vladimir Kolesnikov, sowie Prof. Dr.-Ing. Ahmad-Reza Sadeghi und Prof. Dr. Benny Pinkas für die Betreuung der diesem Papier zu Grunde liegenden Diplomarbeit und Dissertation.

Literatur:

[CKV10] Kai-Min Chung, Yael Kalai, Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In 30. Advances in Cryptology – CRYPTO'10, Vol. 6223 of LNCS, S. 583–501. Springer, 2010.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In 29. Advances in Cryptology – EUROCRYPT'10, Vol. 6110 of LNCS, S. 24–43. Springer, 2010.

[DGKN09] Ivan Damgård, Martin Geisler, Mikkel Krøigård, Jesper B. Nielsen. Asynchronous multiparty computation: Theory and implementation. In 12. International Conference on Practice and Theory in Public Key Cryptography (PKC'09), Vol. 5443 of LNCS, S. 160-179. Springer, 2009. http://viff.dk

[Gen09a] Craig Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford University, September 2009. http://crypto.stanford.edu/craig/

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In ACM Symposium on Theory of Computing (STOC'09), pages 169–178. ACM, 2009.

[GGP10] Rosario Gennaro, Craig Gentry, Bryan Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In 30. Advances in Cryptology – CRYPTO'10, Vol. 6223 of LNCS, S. 465-482. Springer, 2010.

[GH10] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. Cryptology ePrint Archive, Report 2010/520, 2010. http://eprint.iacr. org

[GQ10] Damien Giry, Jean-Jacques Quisquater. Cryptographic Key Length Recommendation, 2010. http://keylength.com.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella. Fairplay – a secure two-party computation system. In 13. USENIX Security Symposium (Security'04). S. 287-302. USENIX, 2004. http://fairplayproject.net

[OPJM10] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, Boaz Moskovich. SCiFI - A System for Secure Face Identification. In 31. IEEE Symposium on Security & Privacy (S&P'10), S. 239–254. IEEE, 2010.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In 17. Advances in Cryptology – EUROCRYPT'99, Vol. 1592 of LNCS, Springer, 1999.

[SV10] Nigel P. Smart, Fre Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In 13. Public Key Cryptography (PKC'10), Vol. 6056 of LNCS, S. 420–443. Springer, 2010.

[Val76] Lesslie G. Valiant. Universal circuits (preliminary report). In 8. ACM Symposium on Theory of Computing (STOC'76). S. 196-203. ACM, 1976.

[Yao86] Andrew C. Yao. How to generate and exchange secrets. In 27. Symposium on Foundations of Computer Science (FOCS'86), S. 162-167. IEEE, 1986.

[Zeit10] Datendiebstahl bei der ZEIT, Meldung vom 11. Oktober 2010 auf Heise Security http://www.heise.de/security/meldung/Datendiebstahl-bei-der-ZEIT-Update-1105127.html

[1] Thomas Schneider. Practical secure function evaluation. Diplomarbeit, Universität Erlangen-Nürnberg, 27. Februar 2008. http://thomaschneider.de/theses/da/

[2] Vladimir Kolesnikov, Thomas Schneider. A practical universal circuit construction and secure evaluation of private functions. In 12. International Conference on Financial Cryptography and Data Security (FC'08), Vol. 5143 of LNCS, S. 83-97. Springer, 2008. http://thomaschneider.de/FairplayPF

[3] Vladimir Kolesnikov, Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In 35. International Colloquium on Automata, Languages and Programming (ICALP'08), Vol. 5126 of LNCS, S. 486-498. Springer, 2008.

[4] Benny Pinkas, Thomas Schneider, Nigel P. Smart, Stephen C. Williams. Secure two-party computation is practical. In 15. Advances in Cryptology – ASIACRYPT 2009, Vol. 5912 of LNCS, S. 250-267. Springer, 2009.

[5] Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, Thomas Schneider. Embedded SFE: Offloading server and network using hardware tokens. In 14. International Conference on Financial Cryptography and Data Security (FC'10), Vol. 6052 of LNCS, S. 207-221. Springer, 2010.

[6] Kimmo Järvinen, Vladimir Kolesnikov, Ahmad-Reza Sadeghi, Thomas Schneider. Garbled circuits for leakage-resilience: Hardware implementation and evaluation of one-time programs. In 12. International Workshop on Cryptographic Hardware and Embedded Systems (CHES'10), Vol. 6225 of LNCS, S. 383-397. Springer, 2010.

[7] Ahmad-Reza Sadeghi, Thomas Schneider, Marcel Winandy. Token-based cloud computing – secure outsourcing of data and arbitrary computations with lower latency. In 3. International Conference on Trust and Trustworthy Computing (TRUST'10) – Workshop on Trust in the Cloud, Vol. 6101 of LNCS, S. 417-429. Springer, 2010.

[8] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In 8. International Conference on Cryptology And Network Security (CANS'09), Vol. 5888 of LNCS, S. 1-20. Springer, 2009.

[9] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, Immo Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. In 17. ACM Conference on Computer and Communications Security (ACM CCS'10), S. 451-462. ACM, 2010. http://tastyproject.net

[10] Ahmad-Reza Sadeghi, Thomas Schneider, Immo Wehrenberg. Efficient privacy-preserving face recognition. In 12. International Conference on Information Security and Cryptology (ICISC'09), Vol. 5984 of LNCS, S. 229-244. Springer, 2009.

[11] Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, Thomas Schneider. Secure evaluation of private linear branching programs with medical applications. In 14. European Symposium on Research in Computer Security (ESORICS'09), Vol. 5789 of LNCS, S. 424-439. Springer, 2009.

[12] Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Annika Paus, Ahmad-Reza Sadeghi, and Thomas Schneider. Efficient privacy-preserving classification of ECG signals. In 1st IEEE International Workshop on Information Forensics and Security (IEEE WIFS'09), S. 91-95. IEEE, 2009.

[13] Ahmad-Reza Sadeghi, Thomas Schneider. Verschlüsselt Rechnen: Sichere Verarbeitung verschlüsselter medizinischer Daten am Beispiel der Klassifikation von EKG-Daten. In Workshop Innovative und sichere Informationstechnologie für das Gesundheitswesen von morgen (perspeGKtive'10), Vol. P-174 of LNI, S. 11-25, GI 2010.

[14] José B. Almeida, Endre Bangerter, Manuel Barbosa, Stephan Krenn, Ahmad-Reza Sadeghi, Thomas Schneider. A certifying compiler for zero-knowledge proofs of knowledge based on sigma-protocols. In 15. European Symposium on Research in Computer Security (ESORICS'10), Vol. 6345 of LNCS, S. 151-167. Springer, 2010. http://zkc.cace-project.eu