Gesicherte Zeitübertragung im Internet of Things mit dem NTS-Protokoll

Martin Langer, Rainer Bermbach

Ostfalia Hochschule für angewandte Wissenschaften, Wolfenbüttel
{mart.langer, r.bermbach}@ostfalia.de

26. September 2017

Abstract

Dieses Dokument beschäftigt sich mit der Zeitsynchronisation von *Internet of Things*-Netzen durch das *Network Time Protocol* (NTP) sowie der Sicherung der Zeitdaten mit dem in Entwicklung befindlichen *Network-Time-Security*-Protokoll (NTS). Der Fokus liegt insbesondere auf der erreichbaren Synchronisationsgenauigkeit und dem zusätzlichen Leistungsaufwand, der zur Sicherung der Zeitinformationen anfällt. Entsprechende Messungen mit einer NTS-Implementierung auf Einplatinencomputern[1] bieten vorläufige Ergebnisse, die einer ungesicherten Kommunikation gegenübergestellt werden und Gegenstand weiterer Diskussionen sind.

Keywords: Network Time Security (NTS); Network Time Protocol (NTP)

1 Einleitung

Das *Internet of Things* (IoT) ist ein wachsender Markt und zieht immer mehr in private und industrielle Bereiche ein. Die komplexe und skalierende Vernetzung der Geräte macht es daher oft unerlässlich, eine zeitliche Synchronisation untereinander aufrecht zu erhalten, um bestimmte Aufgaben oder Funktionen zu tätigen. Betroffen sind somit IoT-Geräte, die vornehmlich auf Einplatinencomputern laufen und Zeitinformationen für ihre Aufgaben nutzen.

[1] Verwendung unterschiedlicher Modelle der Raspberry-Pi-Serie

Ein einfaches Beispiel für die Notwendigkeit genauerer Uhrzeiten stellt Abbildung 1 dar, in der einige IoT-Geräte lokal über ein Router (bzw. Gateway) vernetzt sind und miteinander interagieren. Ein externes System fragt gelegentlich einige Geräte nach Ereignissen ab und speichert diese in eine Datenbank. In einem solchen Fall kann es zu einer inkonsistenten Datenbank führen, wenn die IoT-Geräte unterschiedliche Zeiten aufweisen und folglich den erfassten Ereignissen falsche Zeitstempel zuordnen. Ein Zusammenhang der Ereignisse auf Basis der Zeitstempel in der Datenbank ist nur noch schwer oder gar nicht mehr herstellbar. Darüber hinaus erschwert dieses Szenario auch Aktionen, die in zeitlicher Abhängigkeit zwischen den Geräten stattfindet (z.B. simultane Messungen/Steuerungen). Begegnen kann man diesem Problem durch zeitliche Synchronisierung der Geräte, in dem Zeitsynchronisierungsprotokolle, wie das stark verbreitete *Network Time Protocol* (NTP), zur Anwendung kommen. Hierbei ist jedoch zu beachten, dass je nach Applikation die Manipulation der übertragenden Zeitinformationen durch einen Angreifer zu verhindern ist.

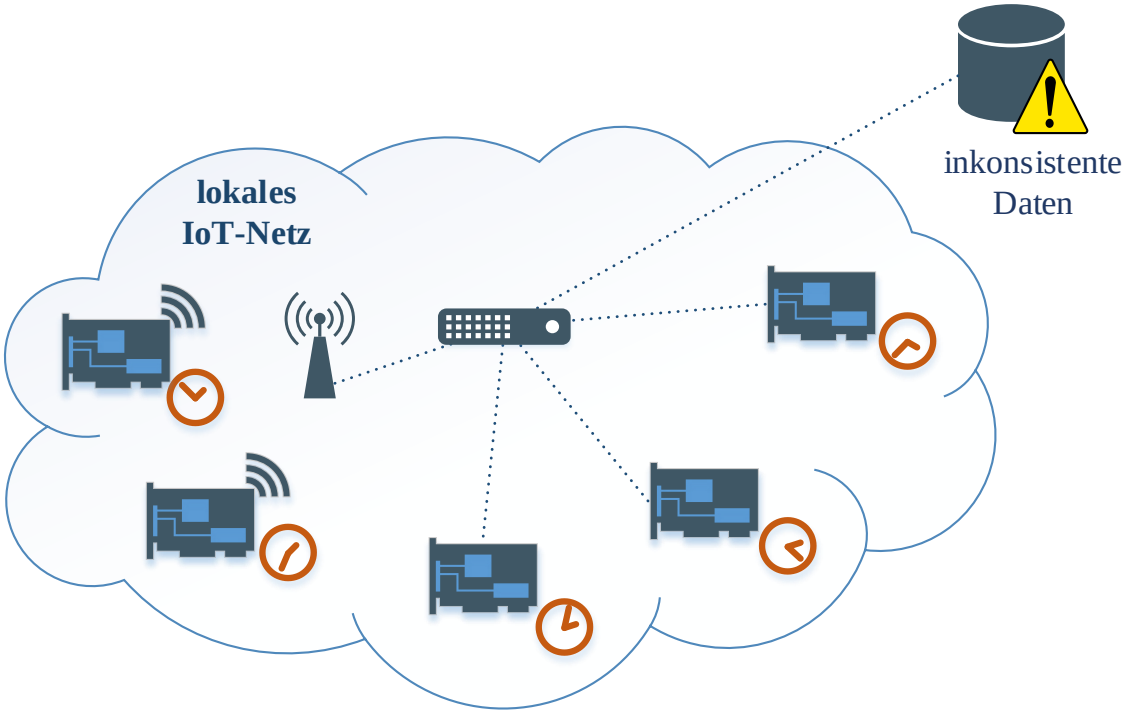

Abbildung 1: Lokales IoT-Netz ohne zeitliche Synchronisation

2 Überblick über das Network Time Protocol (NTP)

Bei NTP handelt es sich um eines der ältesten Zeitsynchronisierungsprotokolle, das weltweit stark etabliert ist. Das im Jahre 1985 von David L. Mills entwickelte Protokoll wurde als RFC 958 [1] veröffentlicht und steht seit 2010 in der aktuellen Version 4 [2] zur Verfügung. Das Protokolldesign ist auf paketbasierte Kommunikationsnetze ausgerichtet und verwendet das verbindungslose UDP-Transportprotokoll, um die entsprechenden Zeitinformationen zu übertragen. Hierbei kommt ein Client/Server-Modell in Verbindung mit einer hierarchischen Struk-

tur zum Einsatz (vgl. Abbildung 2), welches die variablen Paketlaufzeiten bestmöglich ausgleicht. Die Zeitserver werden entsprechend der Schicht einem *Stratum* zugeordnet, das die Qualität der bezogenen Uhrzeit widerspiegelt. Ein Stratum von 0 gleicht hierbei dem Zeitnormal und repräsentiert beispielsweise eine Atomuhr. Bei jeder weiteren Verteilungsinstanz (Zeitserver), die sich in der Kette zwischen dem Stratum-1-Zeitserver und dem NTP-Client befindet, erhöht sich dieses Stratum jeweils um eins.

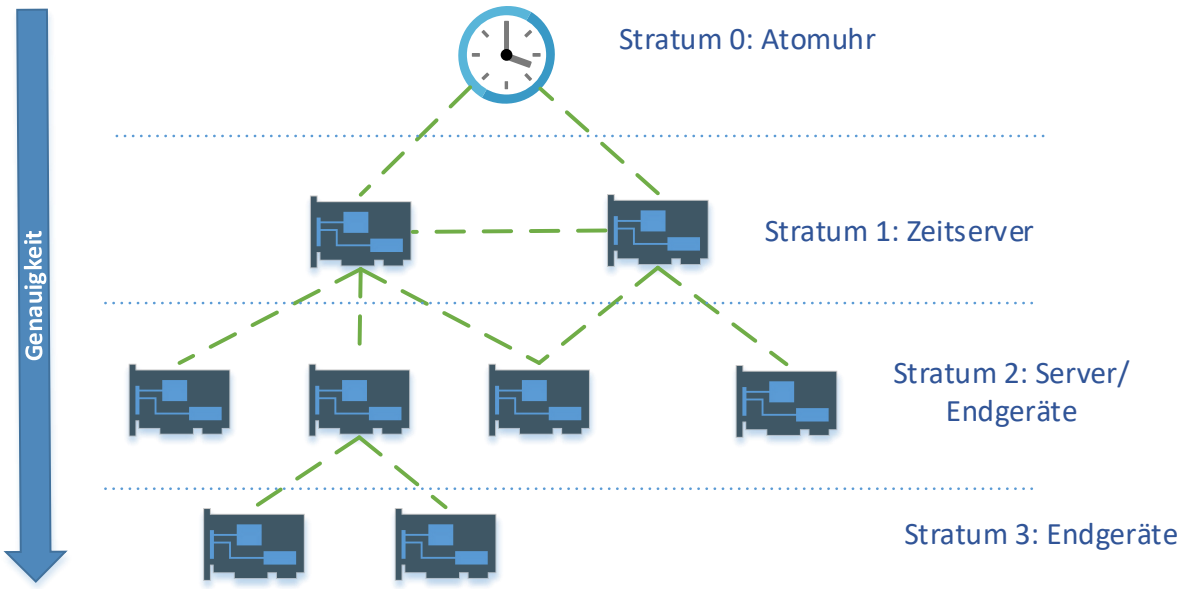

Abbildung 2: Hierarchischer Aufbau von NTP-Teilnehmern

2.1 Allgemeine Funktionsweise

NTP nutzt unterschiedliche Betriebsmodi, welche die Art der verwendeten Kommunikationen zwischen den NTP-Teilnehmern bestimmt. Im symmetrischen Modus kann ein NTP-Server sowohl die Zeit verteilen, als auch empfangen. Dadurch garantiert der Zeitserver seine eigene Synchronität durch den Abgleich mit Zeitservern gleicher Strata. Des Weiteren existiert ein Broadcast-Modus, in dem der Server periodisch Zeitinformationen in einem Netzwerk aussendet, die bei Bedarf von den angebundenen Clients zur Synchronisierung genutzt werden können. Darüber hinaus kommt auch der Client/Server-Modus zum Einsatz, der am häufigsten Verwendung findet und den Unicast-Modus darstellt. In diesem sendet der Client einmalig oder periodisch Zeitanfragen an einen Zeitserver, worauf dieser mit entsprechenden Antworten reagiert und somit die gewünschten Informationen übermittelt [3]. Die hierbei übertragenen Zeitstempel besitzen im NTP-Protokoll eine Länge von 64 Bit, wodurch sich ein Zeitraum von etwa 136 Jahren mit einer Auflösung von 232 Pikosekunden darstellen lässt [4, p. 7]. In Verbindung mit den Paketlaufzeiten im Netzwerk und den daraus resultierenden Fehlern, können durchaus Genauigkeiten von *10 ms* im WAN und im lokalen Netzwerk unter *1 ms* erreicht werden[2].

[2] Je nach Auslastung des Netzwerks und Genauigkeit des lokalen Zeitgebers (Quarz), sind auch Genauigkeiten im unteren zweistelligen Mikrosekundenbereich möglich.

Die Zeitsynchronisation zwischen NTP-Client und Zeitserver funktioniert im Unicast-Betrieb wie folgt. Der Client sendet eine Anfrage in Form eines NTP-Pakets an den Zeitserver. Dieser verarbeitet das Paket, tauscht die IP-Adressen und einige Felder aus und sendet dieses zurück. In dieser Abfolge ergeben sich vier Zeitstempel (vgl. Abbildung 3), wovon die ersten drei Zeitstempel (T_1 bis T_3) im NTP-Paket gespeichert werden. Folgende Zeitstempel werden hierbei erfasst:

- T_1: Clientseitiger Absendezeitpunkt der NTP-Nachricht (Request)
- T_2: Zeitpunkt, zu dem der Server die Nachricht empfangen hat
- T_3: Serverseitiger Absendezeitpunkt der NTP-Nachricht (Response)
- T_4: Zeitpunkt, zu dem der Client die Nachricht empfangen hat

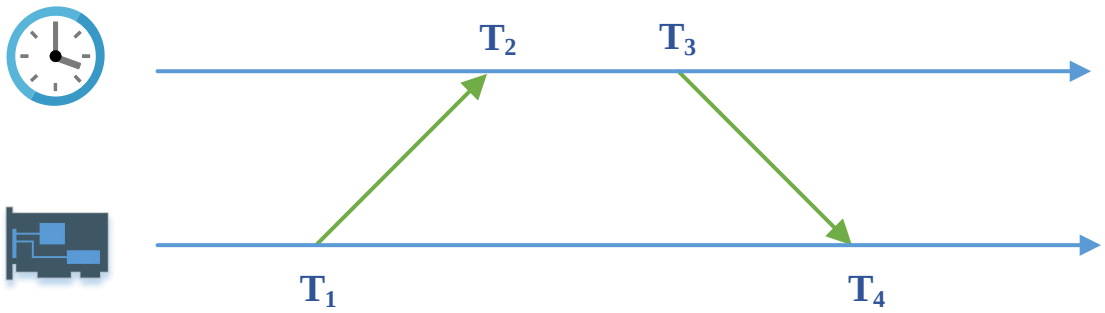

Abbildung 3: Zeitstempel im NTPv4-Protokoll

Aus diesen Zeiten kann der NTP-Client nun den Delay (δ) und den Offset (θ) berechnen. Der Delay stellt die Rundlaufzeit des NTP-Paketes ohne die Rechenzeit dar. Es ist also die reine Zeit, in der sich das Paket im Netzwerk befand. Der Delay berechnet sich gemäß der folgenden Gleichung 1 [2]:

$$\delta = (T_4 - T_1) - (T_3 - T_2) \quad (1)$$

Gleichung 1: Berechnung des Delays in NTP

Im NTP-Protokoll wird davon ausgegangen, dass die Übertragungsdauer der Pakete in beide Richtungen identisch ist. Abweichungen aufgrund von Asymmetrie fließen als Fehler in den Offset mit ein. Dieser berechnet sich entsprechend der Gleichung 2:

$$\theta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \quad (2)$$

Gleichung 2: Berechnung des Offsets in NTP

Der Offset ist die Zeitspanne, um die die Uhrzeiten von Client und Server differieren. Dieser ist die Mittelung der Laufzeitunterschiede der Anfrage- und Antwortnachricht im Netzwerk. Die Rechenzeiten fließen in diese Berechnung nicht mit ein. Somit ist die Genauigkeit bei der Synchronisierung der lokalen Uhrzeit vom Offset abhängig, der daher möglichst gering (im Idealfall gleich 0) ausfallen sollte.

2.2 Probleme bisheriger Sicherheitsmechanismen

Ein immer größer werdendes Problem ist bei NTP die Tatsache, dass die Übertragung der Zeitdaten im Regelfall ungesichert erfolgt. Angreifern ist es somit problemlos möglich, diese Daten einzelner oder aller Geräte zu manipulieren[3] [5] und damit die Kompromittierung des IoT-Netzwerkes zu bewirken (vgl. Abbildung 4). Um die Sicherheit zu erhöhen, wurde bereits im Jahre 1992 das Symmetric-Key-Verfahren als Bestandteil des NTPv3-Protokolls im RFC 1305 [6] eingeführt. Dieses bietet durch die Nutzung von kryptografischen Funktionen die Möglichkeit, die Authentizität der NTP-Teilnehmer zu prüfen und gewährleistet den Integritätsschutz der übertragenen Zeitinformationen. Obwohl dieses Verfahren nach wie vor als sicher gilt, kommt dieses kaum zum Einsatz, da die benötigten Schlüssel auf postalischem Wege ausgetauscht werden müssen. Das ist aufwändig und für skalierende Systeme wie IoT-Netze völlig ungeeignet.

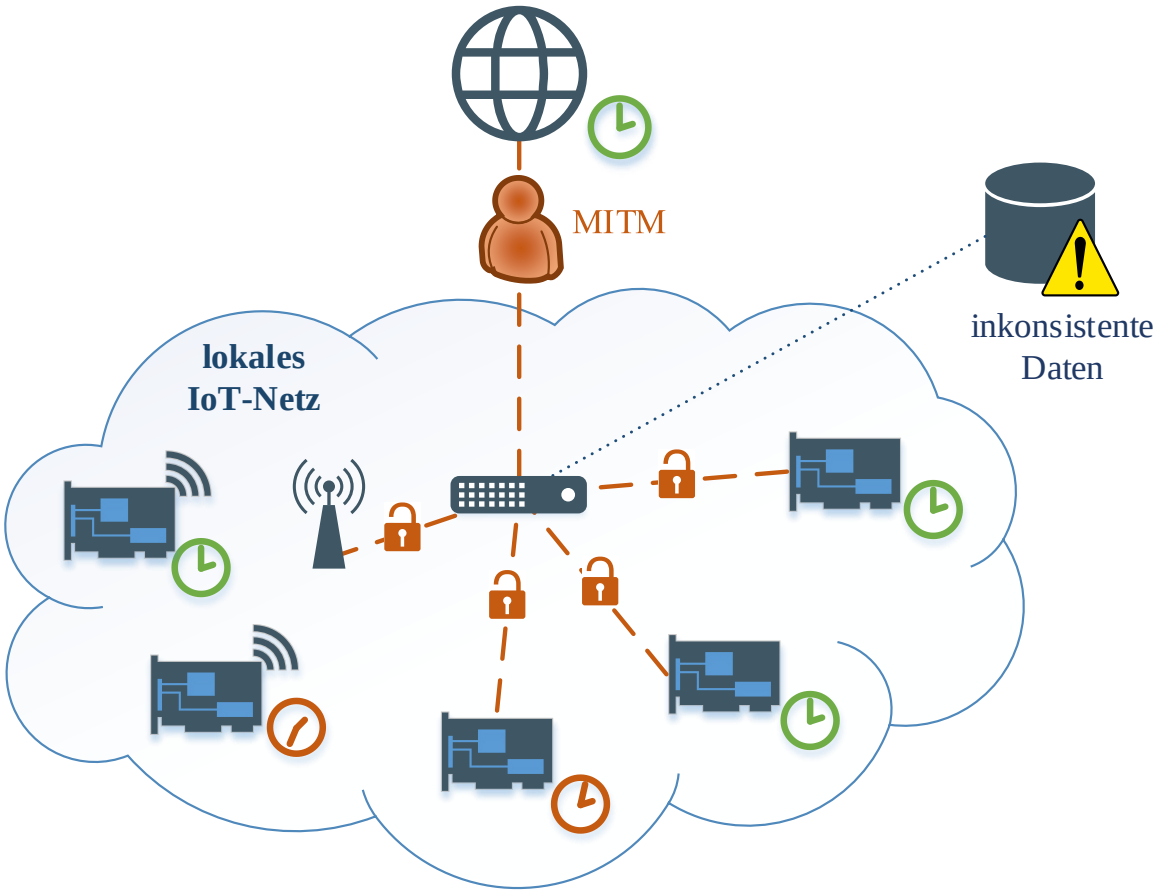

Abbildung 4: MITM-Angriff auf das IoT-Netz

[3] Eine umfangreiche Auflistung der bekannten Angriffe ist im RFC 7384 [10] definiert.

Eine Alternative zum Symmetric-Key-Verfahren bietet die Autokey-Methode, welche im NTPv4-Protokoll verfügbar ist und im RFC 5906 [7] zusätzlich zum Symmetric-Key-Verfahren definiert wurde. Autokey bietet gegenüber dem Symmetric-Key-Verfahren den Vorteil, dass die Schlüssel nicht vorab ausgetauscht werden müssen und so mit der Anzahl der NTP-Nutzer skaliert. Eine Untersuchung der Autokey-Methode [8] zeigte allerdings schwere Sicherheitsmängel, da Angreifer eine Verbindung in wenigen Sekunden brechen können. Vom Einsatz des Autokey-Verfahrens ist daher abzusehen, da es als unsicher gilt.

Ein weiterer Lösungsansatz, die Zeitdaten sicher zu übertragen, bietet eine TLS-Verbindung. Das TLS-Protokoll in der aktuellen Version 1.2 ermöglicht zwar die sichere Übertragung von Zeitdaten, in dem NTP-Pakete durch den gesicherten Kanal gesendet werden, bringt jedoch zwei wesentliche Nachteile mit sich. Zum einen erfordert dies eine dauerhafte TLS-Verbindung, die sowohl für Client, insbesondere aber für den Server ressourcenlastig ist. Zum anderen ist TLS nicht dafür ausgelegt, Zeitdaten zu übertragen, wodurch es zu Einbußen bei der Synchronisationsgenauigkeit beim Client kommt. Daher eignet sich auch dieses Verfahren nicht im IoT-Bereich, in dem die Performance der einzelnen Geräte sehr begrenzt sein kann.

3 Das Network-Time-Security-Protokoll (NTS)

Eine Lösung bietet das Network-Time-Security-Protokoll (NTS) [9], welches eine Erweiterung bestehender Zeitprotokolle[4] darstellt und die gesicherte Zeitübertragung in Computernetzwerken ermöglicht. Dieses befindet sich derzeit im Entwicklungsstatus, steht allerdings kurz vor seiner Fertigstellung und Veröffentlichung als RFC[5]. Die aktuelle NTS-Entwurfsversion ist NTS4NTP-09 [10], wobei sich die hier diskutierte NTS-Version auf die drei Entwurfsdokumente NTS4NTP-06 [11], NTS-15 [12] und CMS4NTS-06 [13] bezieht, da hierfür bereits eine lauffähige Implementierung vorliegt. Das NTS-Protokoll verfolgt unter anderem die Ziele der guten Skalierbarkeit und der Kompatibilität [14]. Somit können dynamisch viele Clients NTS nutzen, ohne vorab einen Schlüsselaustausch mit dem Zeitserver vorzunehmen. Darüber hinaus können auch Teilnehmer geschützte Zeitnachrichten verarbeiten, die kein NTS beherrschen und somit auf die Schutzfunktionen verzichten.

Das NTS-Protokoll ist so gestaltet, dass es Nachrichten in bestehende Zeitprotokolle einbettet, ohne dass eine Anpassung der Spezifikationen solcher Zeitprotokolle notwendig ist. Hierbei funktioniert NTS nach dem Request-Response-Prinzip und verwendet dabei zustandsbehaftete Clients und zustandslose Server. Somit halten Client und Server keine aktive Verbindung zueinander und die Netzwerkbelastung sinkt. Ein Zeitserver kann daher Anfragen eines Clients verarbeiten, ohne dass dieser Informationen über seine Clients speichern muss. Die hierbei übertragenen Parameter werden zwischen Client und Server in verschiedenen Nachrichtentypen ausgehandelt. In einer sogenannten *Verhandlungsphase*, die einmalig zu Beginn einer Kommunikation durchlaufen wird, werden kryptografische Algorithmen festgelegt, Zertifikate ge-

[4] Beispielsweise die Protokolle NTP und PTP (Precision Time Protocol)

[5] *Request for Comments* (RFC) sind Internetstandards, die von der *Internet Engineering Task Force* (IETF) gehalten werden

tauscht und Schlüssel generiert. Zudem erfolgt hier die Prüfung des Zeitservers durch den Client, der die Authentifizierung und Autorisierung durch die *Public Key Infrastructure* (PKI) vornimmt. Sind entsprechende Parameter ausgehandelt, so können die Teilnehmer fortan gesicherte Zeitnachrichten übertragen. Diese werden durch einen *Message Authentication Code* (MAC) geschützt, der mit einem digitalen Fingerabdruck gleichzusetzen ist und jegliche Manipulation nachweisbar macht.

3.1 Kommunikationsablauf

Die Kommunikation beginnt bei NTS durch das clientseitige Versenden einer *Access*-Nachricht. Diese Nachrichten ermöglichen dem Client das Abschicken der folgenden Serveranfragen und schützen den Zeitserver vor sogenannten Verstärkungsattacken. Im Anschluss folgen die *Assoziationsnachrichten*, in denen sich Client und Server über Verbindungsparameter (z.B. kryptografische Algorithmen) einigen. Zusätzlich kann durch Austausch der Zertifikate die Authentizität des Zeitservers durch den Client überprüft werden. Die nachfolgenden *Cookie*-Nachrichten dienen dem verschlüsselten Austausch eines geheimen Schlüssels (das sog. Cookie) vom Server zum Client. Der Server errechnet es aus dem Clientzertifikat und den ausgehandelten Verbindungsparametern. Das Cookie ermöglicht die gesicherte Zeitübertragung und kann jederzeit durch den Server – durch Ändern eines Parameters – verworfen werden. Ist das Cookie im Besitz des Clients, so gilt die *Verhandlungsphase* als abgeschlossen und der Client beginnt mit der *Zeitsynchronisierungsphase*. In dieser sendet er eine gesicherte Zeitanfrage an den Zeitserver. Dabei handelt es sich um eine normale Zeitnachricht (z.B. NTP-Paket), die allerdings mit Hilfe des ausgetauschten Cookies durch einen MAC geschützt ist. Dieser garantiert die Integrität der Nachricht und die Authentizität des Clients. Durch Parameter, die in dieser Zeitnachricht transportiert werden, kann der Zeitserver nunmehr die Echtheit und Unversehrtheit der empfangenen Nachricht prüfen, indem er erneut das Cookie erzeugt und damit den MAC prüft. Ist die Nachricht unverändert, generiert er seinerseits eine gesicherte Zeitnachricht nach dem gleichen Schema und überträgt diese an den Client. Dieser kann nach der Prüfung des digitalen Fingerabdrucks die Zeitinformationen zur lokalen Synchronisation heranziehen. Dieser Vorgang kann dabei sooft wiederholt werden, wie das Cookie gültig ist. Verfällt dieses hingegen, so sendet der Client wieder eine neue Cookie-Nachricht an den Server.

3.2 *Delay Attack* gegen NTS und Gegenmaßnahmen

Wenngleich NTS gute kryptografische Mittel einsetzt, um die Integrität der Zeitnachrichten und die Authentizität der Absender zu garantieren, so bestehen weiterhin sicherheitstechnische Herausforderungen. Ein kritisches Szenario stellt dabei der Delay-Angriff dar [15], in dem ein Angreifer Zeitnachrichten abfängt und verzögert weitersendet. Das NTPv4-Protokoll ist so konzipiert, dass es von symmetrischen Paketlaufzeiten ausgeht. Eine Antwortnachricht vom Zeitserver sollte daher genau solange im Netz sein, wie die Clientanfrage zum Zeitserver. Asymmetrien, wie sie auch im Normalfall in geringen Maßen vorhanden sind, verfälschen somit die lokale Uhrzeit beim Client. Das Ausmaß der Verfälschung ist dabei von der Paketumlaufzeit (δ) abhängig, wobei sich die maximale Zeitabweichung entsprechend Gleichung 3 ergibt:

$$\theta_{\mathrm{Max}} = \frac{(T_4 - T_1) - (T_3 - T_2)}{2} = \frac{\delta}{2} \qquad (3)$$

Gleichung 3: Maximaler Fehler von Zeitnachrichten

Um eine bestimmte Zeitgenauigkeit zu garantieren, ist daher die maximale Paketlaufzeit festzulegen, die noch eine Verarbeitung von Zeitnachrichten erlaubt. Wird beispielsweise eine Obergrenze von 50 ms angesetzt, so liegt die garantierte Genauigkeit eines synchronisierten Clients bei 25 ms, auch wenn die tatsächliche Genauigkeit im Normalfall sehr weit darunterliegt. Übertragen auf IoT-Netze ist es zudem ratsam, lokale Zeitserver einzusetzen, um die Synchronität innerhalb des Netzes zu verbessern (vgl. Abbildung 5). Da Paketlaufzeiten in lokalen Netzen meist deutlich unter einer Millisekunde liegen, könnte eine garantierte Synchronisationsgenauigkeit der Geräte untereinander von 500 µs erreicht werden. Dies ermöglicht die zeitliche Konsistenz aller Geräte, selbst wenn das lokale Netz als solches durch einen Angreifer um max. 25 ms (in diesem Beispiel) gegenüber dem externen Zeitserver verschoben sein könnte. Um interne Netze zu schützen, sollten in solchen Fällen mehrere interne Zeitserver zur Anwendung kommen, um Ausfallsicherheiten zu bieten.

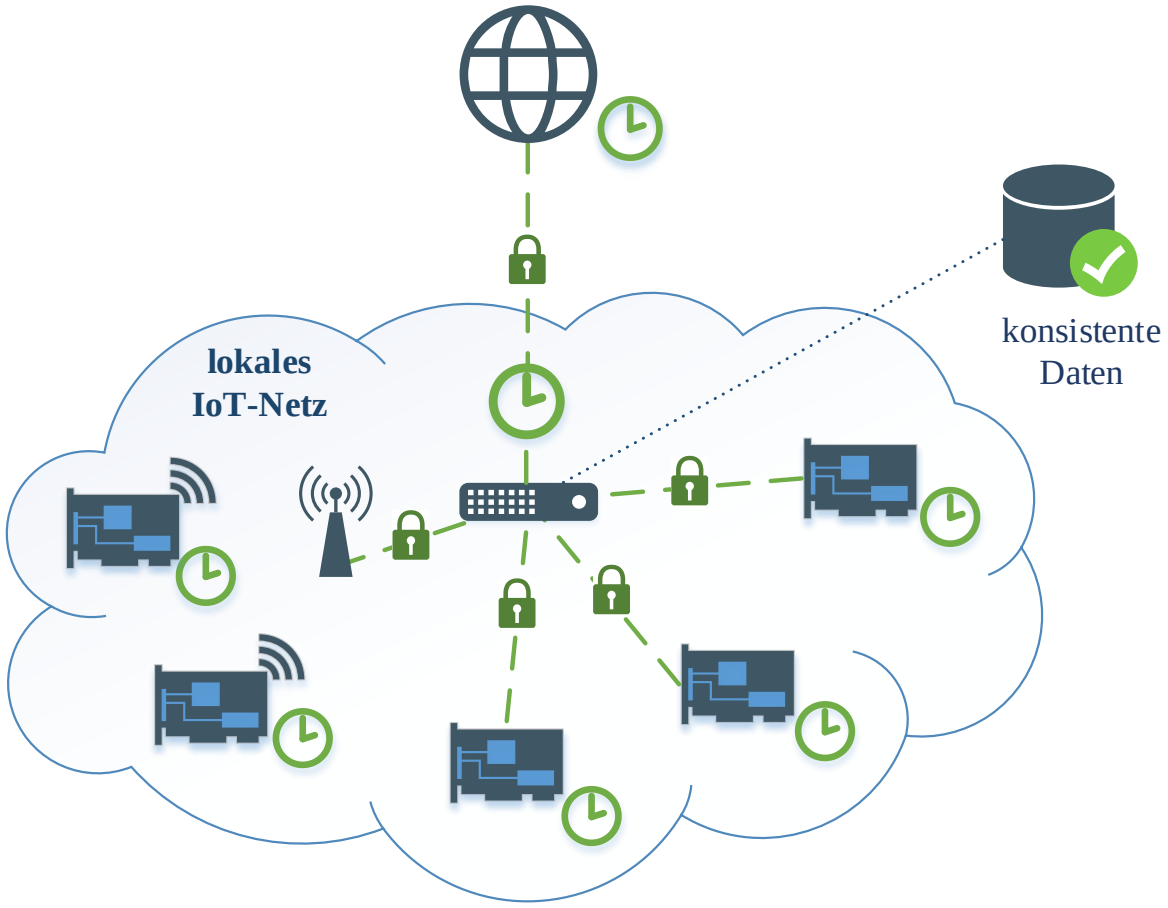

Abbildung 5: Gesichertes und synchronisiertes IoT-Netz

4 Messungen

Dieser Abschnitt stellt bisherige Messergebnisse mit der weltweit ersten NTS-Implementierung[6] dar, um einen Vergleich der gesicherten und ungesicherten Zeitübertragung zu ermöglichen und die realisierbaren Synchronisationsgenauigkeiten im IoT-Netz aufzuzeigen. Als Basis der Messungen dienen *Raspberry Pis* der ersten und dritten Generation, um die Unterschiede verschiedener Leistungsklassen aufzuzeigen. Die typische Dauer einer Messreihe, auf die sich die hier vorgestellten Daten beziehen, beträgt 48-72 Stunden. Die Messungen auf den jeweiligen Geräten verlief simultan, wobei die Hardware und Software der Messobjekte identisch waren. Lediglich die Konfiguration der NTP-Implementierungen variieren, um die Vergleiche zu ermöglichen. Das Betriebssystem, das den Messungen zu Grunde lag, ist das Linux-Debian (Stretch) basierte *Raspbain lite* vom 07.09.2017 [16].

4.1 NTS im Vergleich zur NTP-Referenzimplementierung

Eine wichtige Untersuchung ist die Gegenüberstellung der gesicherten Zeitübertragung zur ungesicherten. Die Werte, die im Folgenden zur Diskussion stehen, entstammen aus den Messauswertungen zweier Messreihen. Die erste basiert auf Messungen mit *Raspberry Pis 1 - Model B* (vgl. Tabelle 1), die zweite auf *Raspberry Pis 3 - Model B* (vgl. Tabelle 2). In diesem Messaufbau befinden sich die Geräte und der verwendete Zeitserver in einem lokalen Netzwerk. Der Zeitserver synchronisiert sich zudem selbst mit den Zeitservern der PTB (z.B. *ptb-time1.ptb.de*) über das Internet. Da die NTS-Implementierung noch kein Teil bekannter NTP-Dienste ist, dient eine eigene NTP-Entwicklung als Träger von NTS. Damit nun ein Vergleich zwischen gesicherter und ungesicherter Zeitübertragung möglich ist, ist die eigene NTP-Lösung ein Teil der Gegenüberstellung. Um Fehlinterpretationen der Tabellen zu vermeiden, werden zunächst die Namenskürzel erläutert, die auch bei den nachfolgenden Messergebnissen in der Tabelle zur Anwendung kommen. **NTPD** ist hierbei ein NTP-Dienst[7], der besonders bei Linux-Systemen genutzt wird und als Referenzimplementierung für ungesicherte Zeitübertragung dient. Das Kürzel **NTP** stellt die eigene NTP-Lösung dar, die auch hier die Zeitdaten ungesichert überträgt. **NTP (NTS)** ist das implementierte Sicherungsprotokoll, dass in Verbindung mit der eigenen NTP-Lösung eine gesicherte Zeitübertragung ermöglicht.

Tabelle 1: Vergleich der NTP-Implementierungen auf dem Raspberry Pi 1

Software	**Delay** [ms]	**Delay Jitter** [ms]	**Zeitabweichung** [ms]
NTPD	0,927	0,017	± 0,050
NTP	1,297	0,117	± 0,100
NTP (NTS)	3,457	0,732	± 0,300

[6] Basierend auf dem NTS-Entwurfsdokument NTS4NTP-06 [11]
[7] Network Time Protocol (NTP) Daemon [17]

Tabelle 2: Vergleich der NTP-Implementierungen auf dem Raspberry Pi 3

Software	**Delay** [ms]	**Delay Jitter** [ms]	**Zeitabweichung** [ms]
NTPD	0,759	0,032	± 0,040
NTP	0,936	0,072	± 0,100
NTP (NTS)	2,643	0,143	± 0,250

Die Tabellen zeigen die Messunterschiede in den Bereichen *Delay*, *Delay Jitter* und *Zeitabweichung* der jeweiligen Software. Das *Delay* ist die von NTP errechnete Paketumlaufzeit im lokalen Netz und berücksichtigt geringe bis mittlere Belastungen des Netzwerks, wobei der Wert den Zentralwert (Median) darstellt. Die typischen Schwankungen, die dabei auftreten, werden durch den *Delay Jitter* (Standardabweichung) repräsentiert, wobei sich das Delay entsprechend um dem Jitter erhöhen kann. Die *Zeitabweichung* stellt die Synchronisationsgenauigkeit des Clients gegenüber dem Zeitserver dar, in dem etwa 99% aller aufgenommenen Messwerte liegen. Dies berücksichtigt auch Quarzschwankungen, die aufgrund von Temperaturänderungen stetig zu kompensieren sind. Bei temperaturstabilen Umgebungen können allerdings Genauigkeiten von 20µs und weniger bei allen Konfigurationen erreicht werden.

Wie die Messauswertungen zeigen, ist die Genauigkeit NTS-gesicherter Zeitnachrichten gegenüber ungesicherten Nachrichten geringer. Gleichzeitig erhöhen sich auch das Delay und der Jitter. Dies ist zwei Einflussfaktoren geschuldet. Einerseits stehen die eigenen Implementierungen (NTP und NTS) erst in einer Version zur Verfügung, in denen noch diverse Optimierungsiterationen zu tätigen sind, andererseits liegt dies am NTS-Design selbst. Hierbei werden NTP-Pakete nach deren Erzeugung und vor ihrer Übertragung durch die kryptografischen Funktionen des NTS-Dienstes gesichert. Die hierfür benötigte Zeit fließt allerdings zusätzlich in das von NTP ermittelte Delay ein. Die Differenzen bei diesen kryptografischen Berechnungen zwischen Zeitserver und Client erzeugen asymmetrische Paketlaufzeiten, die als Fehler in den Offset einfließen (s.a. Abschnitt 2.1) und somit die Synchronisationsgenauigkeit beim Client reduzieren. Zudem ist zu beachten, dass die maximal mögliche Zeitabweichung gegenüber dem Zeitserver die Hälfte der Paketumlaufzeit (Delay/2) entspricht. Auch der Vergleich zwischen der unterschiedlich leistungsstarken Hardware zeigt Differenzen in der Genauigkeit. Stärkere Hardware beschleunigt den Programmfluss und führt somit zu besseren Ergebnissen. Im Falle der optimierten Referenzimplementierung (NTPD), sind allerdings kaum Veränderungen in der Genauigkeit zu erkennen.

Aus diesen Erkenntnissen lässt sich schließen, dass eine NTS gesicherte Zeitübertragung Synchronisationsgenauigkeiten von 0,25 Millisekunden zulässt und eine garantierte Genauigkeit von weniger als 2 Millisekunden ermöglicht. Optimierungen der NTS-Implementierungen könnten eine zusätzliche Verbesserung der Genauigkeit um etwa 50% bewirken[8].

[8] Dieser Wert ergibt sich duch Auswertungen diversen Code-Analysen

4.2 Vergleich unterschiedlicher Kommunikationswege

Die in diesem Abschnitt aufgeführten Messungen veranschaulichen die Paketlaufzeiten und Zeitabweichungen bei der Verwendung gängiger Kommunikationskanäle. Alle Messungen wurden auf dem Raspberry Pi 3 durchgeführt und nutzen dieselbe Software. Tabelle 3 stellt hier die Ergebnisse der ungesicherten NTP-Verbindung dar, wobei die eigene NTP-Implementierung für die Aufzeichnung der Werte verwendet wurde. In Tabelle 4 sind die Werte einer NTS-gesicherten NTP-Verbindung aufgeführt, verwenden ansonsten die identischen Konfigurationen. Der Umfang der Messungen beginnt bei lokalen Netzwerken mit einem lokalen Zeitserver, welcher sich wiederum über das Internet mit den PTB-Zeitservern synchronisiert. Hier erfolgt die Anbindung der Geräte entweder per Ethernet-Kabel (100 Mbit) oder per WLAN (802.11n). Weiter geht es mit der Synchronisierung der Geräte direkt über das Internet mit einem entfernten Zeitserver. Hierfür wurden unterschiedliche Pfade wie Glasfaser (FTTH[9]) oder Kabel-Internet verwendet. Abschließend sind noch Messungen über LTE durchgeführt worden, da dies für abgekoppelte Geräte ohne weitere Netzwerkanbindung von Interesse sein kann.

Tabelle 3: Ungesichertes NTP über verschiedene Kommunikationskanäle (NTP)

Kommunikationskanal	**Zeitserver**	**Delay** [ms]	**Zeitabweichung** [ms]
LAN	Lokal	0,9	± 0,100
WLAN	Lokal	1,5	± 0,150
Internet (Glasfaser)	ptbtime1.ptb.de	30	± 0,800
Internet (Kabel)	ptbtime1.ptb.de	34	± 0,600
LTE	ptbtime1.ptb.de	25 bis 5000	± 600,0

Tabelle 4: NTS-gesichertes NTP über verschiedene Kommunikationskanäle (NTP(NTS))

Kommunikationskanal	**Zeitserver**	**Delay** [ms]	**Zeitabweichung** [ms]
LAN	Lokal	2,6	± 0,250
WLAN	Lokal	3,5	± 0,300
Internet (Glasfaser)	ptbtime1.ptb.de	32	± 0,800
Internet (Kabel)	ptbtime1.ptb.de	36	± 0,600
LTE	ptbtime1.ptb.de	25 bis 5000	± 600,0

Die Messergebnisse zeigen, dass die durch NTS verursachte zusätzliche Zeitabweichung nur in den lokalen Netzwerken zum Tragen kommt und selbst dort keine signifikante Verschlechterung verursacht. Bei Synchronisationen über das Internet sind bereits keine oder nur geringe Unterschiede zwischen einer gesicherten und ungesicherten Zeitübertragung feststellbar. Große Abweichungen zeigten allerdings die Messungen, die über eine LTE-Verbindung liefen, unabhängig davon, ob mit oder ohne NTS. Hier kam es zu starken Schwankungen bei den Paketlaufzeiten, wodurch sich die Zeitabweichung gegenüber den anderen Messungen stark abhob.

[9] FTTH (Fibre To The Home)

4.3 Vergleich der Effizienz

Da in IoT-Umgebungen häufig Hardware mit geringer Prozessorleistung anzutreffen ist, sind auch Untersuchungen der einzelnen Implementierungen auf deren Energieeffizienz sinnvoll. Diese hat Einfluss auf die benötigte Stromaufnahme der Hardware und spielt besonders für batteriebetriebene IoT-Lösungen eine nicht unerhebliche Rolle. Die Tabelle 5 und das dazugehörige Diagramm (vgl. Abbildung 6) stellen die benötigte Prozessorzeit pro Tag dar, die täglich bei den jeweiligen Anwendungen anfällt. Je höher die Werte sind, desto rechenintensiver ist das Programm und umso mehr Energie ist von Nöten. Die Messergebnisse zeigen, dass die eigene NTP-Implementierung derzeit noch mehr Leistung benötigt, als die Referenzimplementierung. Dies liegt zum einen an der vielfach optimierten Referenzimplementierung und zum anderen an dem größeren Poll-Intervall, auf das sich NTPD während der Messungen eingestellt hat. Die eigene Implementierung verwendet im Gegensatz zu NTPD den im RFC angegebenen Algorithmus zur Festlegung des Poll-Intervalls. Dies hat zur Folge, dass bei NTPD weniger Nachrichten[10] verarbeitet wurden und sich somit auch die benötigte Prozessorleistung reduziert. Einen Einfluss auf die Genauigkeit bei der Synchronisierung hat dies jedoch nicht. Ein Vergleich zwischen NTP und NTS zeigt eine weitere Erhöhung der Prozessorzeit bei NTS, die sich insbesondere bei schwächerer Hardware äußert. Auch hier begründet sich die Erhöhung durch fehlende Optimierungsiterationen und den Einsatz von kryptografischen Funktionen, die die Raspberry Pis per Software[11] berechnen müssen.

Tabelle 5: Prozessorzeiten der Implementierungen

CPU-Zeit / Tag	**Raspberry Pi 1** [mm:ss.ms]	**Raspberry Pi 3** [mm:ss.ms]
NTPD	0:28.31	0:09.31
NTP	1:44.51	0:33.38
NTS *(NTP mit NTS)*	2:19.57	0:49.30

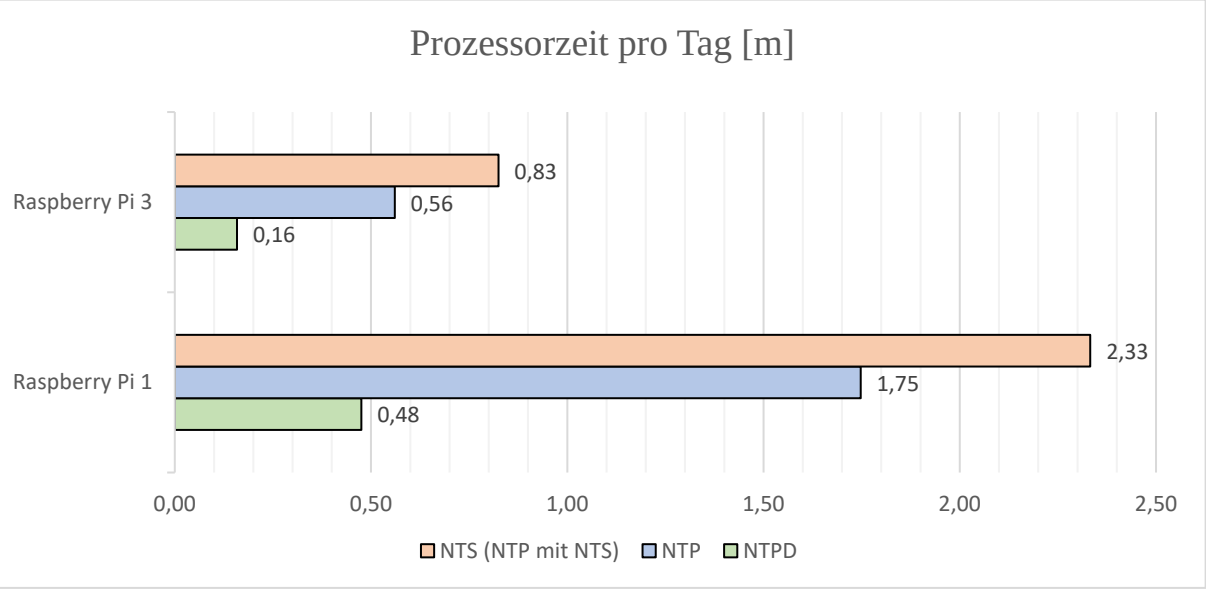

Abbildung 6: Prozessorzeiten der Implementierungen

[10] Verarbeitete Nachrichten am Tag: NTPD: ca. 1230; NTP (mit und ohne NTS): ca. 5400

[11] Kürzere Verarbeitungszeiten kann einige Hardware bieten, die eine AES-Beschleunigung ermöglicht

5 Fazit und weitere Entwicklungen

Die gesicherte Zeitsynchronisation wird gerade im IoT-Bereich eine immer wichtigere Rolle einnehmen, will man die korrekte Interaktion der Geräte bei zeitlich gesteuerten Abläufen sicherstellen. Das Design des Network-Time-Security-Protokolls ermöglicht dies und eignet sich für den Einsatz auf IoT-Geräten, wie die vorläufigen Messungen belegen. Die hierbei erreichten Synchronisationsgenauigkeiten schwanken dabei in Abhängigkeit der verwendeten Netzwerkstruktur und der Anbindungsart der Geräte an das Netzwerk (z.B. kabelgebunden oder Funklösungen) und können durchaus im Mikrosekundenbereich liegen. Die durch NTS eingebrachte Erhöhung der Verarbeitungszeit fällt in den meisten Fällen nicht ins Gewicht. Da das aktuelle NTS-Entwurfsdokument NTS4NTP-09 gegenüber der NTS4NTP-06 massive Designänderungen aufweist, sind neue Messungen sinnvoll. Eine lauffähige Implementierung zu der aktuellen NTS-Version liegt bislang noch nicht vor, befindet sich jedoch parallel zur Spezifikation in Entwicklung. Die Änderungen beziehen sich hierbei auf den Kommunikationsablauf, die Unterstützung weiterer Modi für NTP und den verbesserten Datenschutz. Die Fertigstellung der NTS-Spezifikation wird für Anfang 2018 erwartet.

6 Referenzen

[1] D. Mills, *Network Time Protocol (NTP),* Request for Comments: 958: Internet Engineering Task Force (IETF), Sep. 1985, [Online] URL: https://tools.ietf.org/pdf/rfc958.pdf – Datum des Abrufs: 21.09.2017.

[2] D. Mills; et al., *Network Time Protocol Version 4: Protocol and Algorithms Specification,* Request for Comments: 5905: Internet Engineering Task Force (IETF), Juni 2010, [Online] URL: https://tools.ietf.org/pdf/rfc5905.pdf – Datum des Abrufs: 21.09.2017.

[3] O. Hager, *Allgemeines zum Network Time Protocol,* 2002, [Online] URL: http://doc-tcpip.org/Ntp/basics.html – Datum des Abrufs: 21.09.2017.

[4] D. Mills, *Network Time Protocol Version 4 - Reference and Implementation Guide,* Technical Report 06-6-1 (TR 06-6-1), University of Delaware: NTP Working Group, Juni 2006, [Online] URL: https://www.eecis.udel.edu/~mills/database/reports/ntp4/ntp4.pdf – Datum des Abrufs: 21.09.2017.

[5] A. Malhotra; I. E. Cohen; E. Brakke; S. Goldberg, *Attacking the Network Time Protocol,* Boston University, Boston, MA., 21. Okt. 2015, [Online] URL: https://www.cs.bu.edu/~goldbe/papers/NTPattack.pdf – Datum des Abrufs: 21.09.2017.

[6] D. Mills, *Network Time Protocol (Version 3) - Specification, Implementation and Analysis,* Request for Comments: 1305: Internet Engineering Task Force (IETF), März 1992, [Online] URL: https://tools.ietf.org/pdf/rfc1305.pdf – Datum des Abrufs: 21.09.2017.

[7] B. Haberman; D. Mills, *Network Time Protocol Version 4: Autokey Specification,* Request for Comments: 5906: Internet Engineering Task Force (IETF), Juni 2010, [Online] URL: https://tools.ietf.org/pdf/rfc5906.pdf – Datum des Abrufs: 21.09.2017.

[8] S. Röttger, *Analyse des NTP-Autokey-Verfahrens (Projektarbeit),* Technische Universität Braunschweig, Physikalisch-Technische Bundesanstalt, 09. Sep. 2011, [Online] URL: http://docplayer.org/4552446-Analyse-des-ntp-autokey-verfahrens.html – Datum des Abrufs: 21.09.2017.

[9] ietf.org, *Network Time Security,* [Online] URL: https://datatracker.ietf.org/doc/draft-ietf-ntp-network-time-security – Datum des Abrufs: 21.09.2017.

[10] D. Franke; D. Sibold; K. Teichel, *Network Time Security for the Network Time Protocol (NTS4NTP),* Internet-Draft: draft-ietf-ntp-using-nts-for-ntp-09: Internet Engineering Task Force (IETF), 26. Juni 2017, [Online] URL: https://tools.ietf.org/pdf/draft-ietf-ntp-using-nts-for-ntp-09.pdf – Datum des Abrufs: 20.09.2017.

[11] D. Sibold; S. Roettger; K. Teichel, *Using the Network Time Security Specification to Secure the Network Time Protocol (NTS4NTP),* Internet-Draft: draft-ietf-ntp-using-nts-for-ntp-06: Internet Engineering Task Force (IETF), 22. Sep. 2016, [Online] URL: https://tools.ietf.org/pdf/draft-ietf-ntp-using-nts-for-ntp-06.pdf – Datum des Abrufs: 20.09.2017.

[12] D. Sibold; S. Roettger; K. Teichel, *Network Time Security (NTS),* Internet-Draft: draft-ietf-ntp-network-time-security-15: Internet Engineering Task Force (IETF), 22. Sep. 2016, [Online] URL: https://tools.ietf.org/pdf/draft-ietf-ntp-network-time-security-15.pdf – Datum des Abrufs: 20.09.2017.

[13] Sibold; et al., *Protecting Network Time Security Messages with the Cryptographic Message Syntax (CMS), CMS4NTS,* Internet-Draft: draft-ietf-ntp-cms-for-nts-message-06: Internet Engineering Task Force (IETF), 25. Februar 2016, [Online] URL: https://tools.ietf.org/pdf/draft-ietf-ntp-cms-for-nts-message-06.pdf – Datum des Abrufs: 20.09.2017.

[14] T. Mizrahi, *Security Requirements of Time Protocols in Packet Switched Networks,* Request for Comments: 7384: Internet Engineering Task Force (IETF), Okt. 2014, [Online] URL: https://tools.ietf.org/pdf/rfc7384.pdf – Datum des Abrufs: 21.09.2017.

[15] T. Mizrahi, *A game theoretic analysis of delay attacks against time synchronization protocols,* International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 28 Sep. 2012, [Online] URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.720.6334&rep=rep1&type=pdf – Datum des Abrufs: 21.09.2017.

[16] *Raspbian Stretch Lite,* [Online] URL: http://downloads.raspberrypi.org/raspbian/images/raspbian-2017-09-08/2017-09-07-raspbian-stretch.zip – Datum des Abrufs: 21.09.2017.

[17] Mills, D. L., *Network Time Protocol (NTP) Daemon,* 2005, [Online] URL: https://www.eecis.udel.edu/~mills/ntp/html/ntpd.html – Datum des Abrufs: 21.09.2017.