SSTP: a Scalable and Secure Transport Protocol for Smart Grid Data Collection

Young-Jin Kim, Vladimir Kolesnikov, Hongseok Kim, and Marina Thottan
Bell Labs, Alcatel-Lucent
Murray Hill, NJ, USA
{young.jin_kim, kolesnikov, hongseok.kim, marina.thotthan}@alcatel-lucent.com

Abstract — **Emerging smart grid networks are expected to have massive amounts of data continuously generated from various measuring devices (intelligent sensors, advanced meters, electric vehicle charging stations) which are embedded in the power grid. Data generated from these measuring devices must be delivered securely and reliably to utility control centers for wide-area monitoring and control and to estimate the overall grid status in a timely and precise manner. The collected data is also used for incentivizing consumer participation for improving power stability.**

Transport protocol requirements for such periodic grid measurement data are characterized as *lifetime-lived, secure, and reliable* delivery of *short* flows (usually less than 1.5 KB) over utility-Wide Area Networks (WANs). However, our survey shows that there is no well-known transport protocol that can support the above characteristics in a scalable and light-weight manner. Motivated by this, we design a scalable and secure transport protocol, *SSTP*, exploiting the notion of a "State-token" which is issued with each server message and which is subsequently attached to corresponding client message delivered to the server. Compared with existing well-known transport and security schemes, SSTP enables scalable server deployments as servers do not keep state (for security and communication) per client and thus computation/memory overheads are significantly reduced.

I. Introduction

The limited communication and security capability of non-IP based networks, currently deployed in utility power grids, are crucial impediments to future smart grid deployments aimed at improving energy efficiency, resiliency against power flow disruptions, and reducing carbon emissions by incorporating renewable energy sources. Therefore, the power industry is undergoing a major network transformation by designing a new IP-based smart grid communication network with enhanced security and reliability.

One distinguishing aspect of the smart grid communication network is the large-scale deployment of sensors and smart meters which send periodic updates to the utility control center; e.g., in Manhattan, NY, several million meters must be deployed to cover all customer households. The IP network being deployed is expected to transport the sensor data in a *trustworthy* (secure and reliable) fashion to the utility's control centers, where the data will be used to evaluate the grid state, and current electricity consumption. However, the sensors that generate the data are typically computationally constrained entities. It is in this context that we consider the specific problem of designing transport protocols that can *scalably* and *securely* deliver sensing data with *low overhead* in terms of memory space, computation, and communication bandwidth.

Sensor data collection [1] is an integral part of smart grid communications. The data traffic generated from sensors or meters, which are typically located beyond utility security perimeters, is expected to be carried over a purpose-built utility network. The purpose-built utility network is isolated from the highly critical SCADA-based communication network[1]. It is also isolated from the public Internet[2] because of the significant security and availability issues that are encountered when utility data traffic is multiplexed with Internet data traffic.

The general characteristics of smart grid sensor data can be described as follows. First, data is carried over semi-permanent security and transport associations between sensors and utility-side servers. Second, on the purpose-built utility network, data toward utility (north-bound data), such as meter readings, is periodically collected; in contrast, data originating from the utility (south-bound data), such as management information, is relatively rare. This implies that north-bound data would dominate network resource consumption, as compared with south-bound data. Third, north-bound data must be delivered in a reliable and timely fashion for precise estimation of instability-induced conditions like a large mismatch between actual electricity usage and electricity supply based on day-ahead demand prediction. Comparatively, south-bound data to sensors or meters is not critical for reliable and timely delivery, since this information can be easily overridden by new data. Fourth, secure communication ensuring data credibility is necessary for safe power grid operations. We note that security requirements do not need to be symmetric. For example, authentication is mandatory for delivering the southbound electricity price information, but there is no requirement for confidentiality. In contrast, for the north bound metered data, both authentication and confidentiality are critical. Lastly, field sensors are unlikely to have a full set of operating systems and protocol stacks due to limited computation resources, while utility-side servers have plentiful computing resources to deal with the significant amount of data received from the large number of deployed sensors. E.g., TI MSP 430 microcontroller series [2] adopted in many sensor hardware platforms feature 2-18KB RAM, 1-256KB EPROM, and 8-16MHz CPU: a subset of OS and protocol features can be installed for the platforms; however, due to lack of storage space, data has to be delivered to servers

[1] In this paper, the term, smart grid communication network, does not include the traditional SCADA communication between utility control center and substations that contain mission-critical power equipments. Note that the SCADA communication network is being improved through advanced IP technologies like MPLS for timely monitoring of substation status and safely controlling critical power equipments.

[2] Note that Open Automated Demand Response (OpenADR) driven by National Lawrence Berkeley Lab is to our best knowledge the only known proposal replying on a public network. OpenADR documents outline technology neutral communications specification and data models using Internet-based Web Services to send demand response signal to end-use customer systems (http://openadr.lbl.gov).

in a utility network. To the best of our knowledge, there is no published protocol that accommodates above data transport requirements in a scalable and lightweight manner.

To remedy the situation, we propose a scalable and secure transport protocol *SSTP* for smart grid data collection that addresses the above requirements. The key idea of SSTP is that the server does not need to maintain any state (for security and communication) on a per client basis. Our main contributions in this work are: (1) We design an inherent and lightweight security scheme for the SSTP protocol thus removing the costly dependency on support for TLS [3] or IPSec [4], (2) Our proposed *state-token* concept enables scalable stateless server deployments for data collection. One important security and transport benefit of implementing a stateless server is that SYN (the connection request in TCP) flooding from a large number of clients induced by server restarts or failures can be avoided.

To motivate our design consideration, we first begin with a survey of current transport and security protocols, and highlight the necessity for designing novel scalable transport protocols for smart grid data collection. We will subsequently describe the protocol details for the Scalable Secure Transport Protocol (SSTP) along with its performance evaluation.

II. REVIEW OF EXISTING TRANSPORT PROTOCOLS

We now show that none of the existing transport protocols meets the security and reliability characteristics of the smart grid sensor data network in a scalable and lightweight manner.

TABLE I: FEATURES OF TRANSPORT PROTOCOLS.

Features	TCP	SCTP	SSTP
Security Schemes	TLS or IPSec	TLS or IPSec	Inherent built-in
Flooding Attacks	Cookie [5-6]	Cookie [7]	Cookie [6]
Connection Establishment	Three-way handshake	Four-way handshake	Four-way handshake
Duplex Comm.	Yes	Yes	Mostly simplex
Reliable Delivery	ACK with SACK	ACK with SACK	ACK
In-ordered delivery	Mandatory	Optional	None
Sender-side delay	Congestion Control	Congestion Control	None
Receiver-side delay	Delayed ACK	Delayed ACK	None
Flow Control	Receiving window	Receiving window	Sending window
Transmission or Reception Buffer	128KB per connection[3]	128KB per connection	Chosen by applications
Multi-homing/stream	No	Yes	No

Security and Scalability: As shown in Table I, no well-known protocol has inherent data authentication and confidentiality extensions to prevent eavesdropping, tampering, or message forgery. Instead, external mechanisms such as TLS or IPSec are typically employed for ensuring data security. This is not problematic in itself; however this results in significant overhead in terms of negotiation procedures to accommodate low computing capable end devices. In addition, there is a concern as to whether TCP and SCTP can address end-to-end security for large-scaled and lifetime-lived communication environments in a scalable manner. Consider hourly meter readings in New York City. From networking aspect, meters send measured data to either far-side utility control center (UCC) or near-by aggregators from where the data is forwarded to utility UCC (split-aggregation [10]). In the former case, for a given number N of meters, UCC must either permanently keep O(N) states (security info, tx/rx buffer, connectivity info) or process O(N) secure connection requests per hour for meter readings. Neither of these approaches can avoid scalability concerns and single point of bottlenecks/failures in the UCC. On the other hand, the split-aggregation approach may be better with respect to performance and scalability. Unfortunately, this approach compromises the end-to-end confidentiality as it cannot maintain a TLS (or IPSec) session that spans from meters to UCC. Thus, data has to be decrypted at the aggregators and then encrypted again. Additionally, the overhead (memory footprint and CPU load) required for executing TLS along with either TCP or SCTP is expensive for resource-constrained end devices.

Availability and Scalability: Existing well-known connection-oriented transport protocols may incur SYN flooding when a utility-side server associated with a large number of sensors or meters is abruptly restarted or failed. All sensors associated with this server will simultaneously send thousands of connection requests to the server to reestablish their security associations with the utility as soon as possible. It would happen immediately after a restart or a power outage, regardless of the need to send any data to the UCC. Thus, transport protocols for smart grid data collection must address the SYN flooding problem and also be able to scalably manage lifetime-lived associations.

TCP Latency & Reliability: In-order sequenced delivery of TCP can significantly increase latency under frequent packet drop in routers or packet loss over wireless links. TCP receivers buffer those received segments that are higher than the expected sequence number. This prevents the data from being delivered to a corresponding application until a segment having the expected sequence number arrives. This re-sequencing delay in the receiver results in increased end-to-end latency. However in-order delivery is not required for smart grid data transport as data is always time-stamped at the sender side. Note that in-order delivery can be disabled in SCTP, as shown in Table I. Delayed acknowledgement [11] typically enabled by default in TCP Reno or later versions reduces the number of ACKs to be sent-back to senders, and, accordingly, throughput and resource utilization are highly improved. However it can still increase the latency by up to 40ms under periodic data delivery for data sizes that are smaller than the maximum segment size (MSS). Typically, sensor data size is no more than 1KB [12]. For avoiding this receiver-side-induced latency problem, the delayed ACK timeout (in Linux, /proc/sys/net/ipv4/tcp_delack_min) can be reduced by administrators, but this is not a recommended action as manipulating the option can degrade performance [11].

TCP enables reliable delivery by using the cumulative acknowledgement scheme. Thus, it follows that average TCP end-to-end delay [13] is more than one RTT (Round-Trip Time) even under ideal network conditions, where no packet is dropped or lost, and there is no delay caused by flow control due to the large sliding window size on the receiver side. TCP congestion control schemes [14] can prolong delivery latency in the face of packet drop or loss. The sender-side retransmits lost segments after receiving either three duplicated

[3] For ensuring stable performance, 128KB is a default size tuned in Linux distributions. As shown in [8], RAM footprints for TCP and SCTP can be reduced even to 2KB and 12KB respectively. However, as described in [9], reduction to 2KB for TCP stack has to be carefully dealt since it is a trade-off between footprint reduction and performance degradation.

acknowledgements received (fast recovery) or a timeout is expired (slow start). These schemes were devised for avoiding unnecessary retransmissions by discriminating between lost segments and late-arriving segments which took a longer path (it can occur due to IP routing protocols such as OSPF). Thus, the sender-side by design increases the retransmission delay. The TCP congestion control algorithm limits the data injection speed until network recovers from the congestion event. Selective acknowledgement scheme [15] combined with the cumulative acknowledgement scheme deal with latency issues by quickly recovering multiple lost segments. However, the first among recovered segments has no performance gain as its delivery is tried only after the receipt of three duplicate acknowledgements.

Latency, Heavyweightness, & NAT-unfriendliness of SCTP: SCTP was devised to deliver *aggregated* telephony messages between high-powered telecommunication systems with multiple line cards. For avoiding line blocking in one multi-streamed connection, in-order sequenced delivery is disabled in SCTP and thus the buffering delay in the receiver side is reduced. However, SCTP is not completely free of the latency issues of TCP, due to the congestion control and delayed ACK implementation inherited from TCP. Also, it is unlikely that SCTP can run on sensors or meters with limited computing resources (*e.g.*, 10 KB RAM), as even the SCTP lightweight version [8] is relatively heavy in terms of memory usage and CPU load for message processing. Moreover, a recent report [16] shows that the choice of SCTP must be made carefully under environments with NAT (Network Address Translation) traversal. Most currently deployed NAT boxes are not SCTP-friendly. However, due to its high assurance features, SCTP may be a good transport candidate for some smart grid applications, such as substation automation, requiring multi-homed communications between the UCC and the substations

Based on this review we conclude that existing transport protocols are insufficient to address the transport characteristics of the smart grid sensor network. We next discuss design motivation of the SSTP. First, we point out that the smart grid sensor network typically consists of measured data whose size is such that it can be contained in one protocol message without being segmented into multiple chunks. It means that traffic flow-based semantics employed in most known transport protocols are inappropriate in this case. Also, since the data is carried over a purpose-built utility network, packets are rarely lost due to congestion. It means that a congestion control scheme is also not required for the data collection.

III. Design of SSTP

In our approach, we address all inefficiencies of prior protocols and build our presentation in a top-down manner for simplicity of presentation and understanding.

A. Overview of the approach

Our main goal for SSTP is scalability both in terms of security and communications. Due to the special communication patterns of data collection – a large number of clients infrequently communicating with servers, we identify the following avenues for significant optimizations.

Symmetric-key-based protocols: In data collection, clients only talk to the server, and one pre-shared key (PSK) per client will suffice. In this setting, the use of costly public-key credentials will not bring its benefit of system-wide reduction of the number of keys. Since symmetric key operations are hundreds of times faster than public-key ones, SSTP using only symmetric-key operations for all security extensions.[4]

Server's state independent of the number of clients: In data collection such as meter readings, the server must continuously maintain associations with a very large number of clients, who communicate periodically-but-infrequently. Maintaining such associations is very taxing on system resources, if, as is standard in practice and research protocols, the server in fact keeps security and transport information associated with the session (keys, counters, and so on.). Our basic idea is to provide a light-weight (both for client and the server) implementation which encrypts and authenticates the associated session state, and then gives the resulting encryption for the client to temporarily store and returns it to the server with his next message. In this way, a server does not keep session state after sending the encryption back to a client and can quickly restore it when the next message from the client arrives. Note that our idea can avoid extreme overloading in the face of server restart or failures.

Lightweight protocols inherently supporting security: As implicitly shown in Table I, the design principle of SSTP is the achievement of lightweight and secure transport protocol implementation to provide affordable end-to-end solutions for resource-constrained devices which find it hard to accommodate known-but-heavy security and transport schemes.

B. Pre-shared key based authenticated key exchange (AKE)

As is standard in secure communications, we operate with long-term keys and session keys. Long-term keys are long-lived credentials; in our case they are pre-shared keys (PSK) that each client and server had agreed on before they entered the system. PSKs are not used for secure data transport (as it is expensive to replace them when compromised) and instead used as input to AKE, a two-party protocol, which allows participants, upon mutual authentication, to securely and privately determine one session key. The output of the AKE, the session key, is then used for securing data transport between the participants.

Pre-shared key assignment: Following our goal of achieving server storage independent of the number of clients, we describe how a server can efficiently store a large number of clients' long-term keys. Our novel idea is to have these keys not to be truly random, but, rather, pseudo-randomly generated from the server's master key *k*. That is, given a server's master key *k*, for a client with identity *id*, we set its long-term key $k_{id} = AES_k(id)$. The Advanced Encryption Standard (AES) function above can be of course substituted by any suitable pseudorandom function generator (PRFG). Each client with identity *id* is then

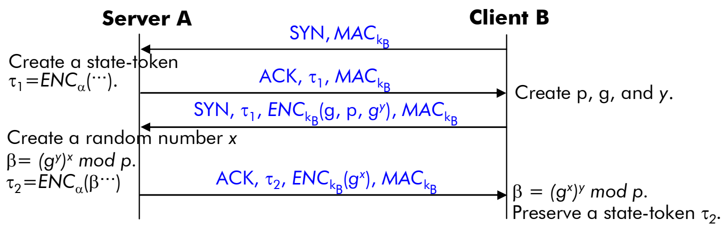
[4] In our presentation, we include the authenticated Diffie-Hellman (DH) key agreement [17] to achieve *perfect forward secrecy* (PFS) – resilience of completed key exchange sessions to possible future PSK compromise. We note that public-key operations of DH can be avoided at the security cost of not achieving PFS. We further stress that SSTP does not preclude the use of public-key-based credentials. If their use is warranted (e.g., in many-to-many communications), and hardware supports this (infrequent) expense, SSTP can be naturally modified to run certificate-based key exchange. In the setting with small server state, our state-token approach is particularly important, as it allows avoiding repeated expensive public-key-based KE.

provisioned, e.g., at the time of manufacture, with k_{id}. The server need not store this key, as he can readily generate it, given his master key and the client's identity. Security properties of PRFG guarantee that none of the client keys can be distinguished from a random string, even if the adversary obtains keys of all other clients. Thus, these keys are safe to use.

Connection Establishment with Diffie-Hellman Exchange: For a client-server pair, a secure transport connection is established through the four-way handshake procedure shown in Figure 1. All messages between clients and servers during connection establishment phase are protected by a message authentication code (MAC), computed with the PSK k_B on the corresponding message (denoted by MAC_{kB}). Also a symmetric session key between a client-server pair is computed by the DH key exchange providing protection against some dictionary attacks and ensuring perfect forward secrecy.

Figure 1 shows client B intending to establish secure transport connection with server A. B sends SYN (the connection request in TCP) message. If server A receives SYN message missing the state-token (the detail of state-token will be discussed next), it creates a new state-token τ_1 and sends back the ACK message with state-token τ_1. When client B gets the ACK message, prime p, generator g, and secret exponent y are randomly generated. Client B sends new SYN message containing state-token τ_1 and a set of numbers (p, g, and exponential g^x) encrypted with PSK k_B. When server B gets the SYN message, it can decrypt the numbers using PSK k_B and compute symmetric session key $\beta = g^{yx}$ mod p from g^y, p, and its own secret random exponent x. (Actually β will be set to be a hash of the DH group element g^{yx}, but we omit this detail for clarity). Conversely, when client B gets ACK message with new state-token τ_2 and its server's encrypted exponential ENC_{kB} (g^x), it can compute the same session key β from g^x, p, and y (B's secret random exponent).

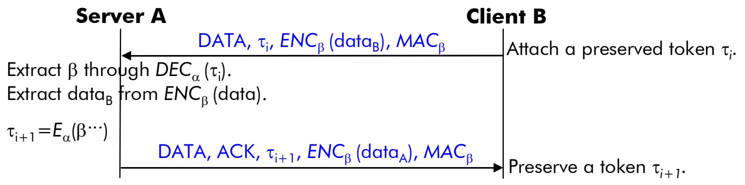
In fact, the SSTP handshake procedure is broadly influenced by TLS DHE_PSK key exchange [18] and TCP cookie transaction [6]. The main difference is that servers preserve neither transport nor security state for clients through the use of state-token to be discussed next. Furthermore, compared with alternatives, SSTP performs connection establishment procedure combined with an AKE and thus overheads can be reduced.


Server A
Client B
SYN, MAC_{k_B}
Create a state-token
$\tau_1 = ENC_\alpha(\cdots)$.
ACK, τ_1, MAC_{k_B}
Create p, g, and y.
SYN, τ_1, $ENC_{k_B}(g, p, g^y)$, MAC_{k_B}
Create a random number x
$\beta = (g^y)^x \bmod p$.
$\tau_2 = ENC_\alpha(\beta\cdots)$
ACK, τ_2, $ENC_{k_B}(g^x)$, MAC_{k_B}
$\beta = (g^x)^y \bmod p$.
Preserve a state-token τ_2.
Figure 1: SSTP Connection Establishment with AKE.

C. State-token and Secure Communications

We now present the punch line of our approach, namely, the technique to effectively offload the state associated with the client to the client himself. At a high level, the idea is to encrypt and authenticate the state, and store it at the client. We realize it as illustrated in Figures 1 and 2. First, the server will maintain a set of separate symmetric keys tk_{id}, securely derived from each client's id (e.g., $tk_{id} = AES_k$ *("tokenkey",id)*), for the purpose of this encryption/authentication. Further, upon being about to send ACK message to a client, the server encrypts and authenticates with tk_{id} the state (session key, time stamp, client's id, and counter) of our protocol's association, and thus obtains a *state-token.* This state token is then sent to the client and then erased from the server together with session state. Client, upon receipt of the token, stores it, and includes it verbatim in his next message to the server. Note that the most-recent preserved state-token can only be appended to new messages but not to retransmission messages which must include old state-tokens they originally included. We emphasize that the (encrypted) state token is not further encrypted, in contrast with the rest of the client's message, so that server can decrypt the state token and restore the session state. Upon receipt of the client's message, the server first extracts, decrypts, and verifies integrity of the state token. Upon successful verification, the server restores the session state, and processes the client's message as usual. Again, upon termination of the message processing, the server repeats the above procedure.

On the client side, messages are encrypted and authenticated using a session key β computed by the AKE. In the server side, if the state-token verification passes, we can extract the following from the state-token using a key tk_{id} : session key β, token issue time TS_i, client's id *id*, and counter, N_i. Then, the message can be further processed if all the following conditions hold: (1) the source of message equals to *id*, (2) the message's sequence number is greater than or equal to N_i, and (3) TS_i is less than current time and not beyond an age limit.


Server A
Client B
DATA, τ_i, ENC_β ($data_B$), MAC_β
Attach a preserved token τ_i.
Extract β through DEC_α (τ_i).
Extract $data_B$ from ENC_β (data).
$\tau_{i+1} = E_\alpha(\beta\cdots)$
DATA, ACK, τ_{i+1}, ENC_β ($data_A$), MAC_β
Preserve a token τ_{i+1}.
Figure 2: SSTP Data Transfer.

D. Reliability and Latency

For scalability, a server has one receiving buffer across all its clients and also no sending buffer. Recall that sensor data from clients is continuously being sent to a server, while management commands are rarely sent by the server to clients. A client can send a new message immediately after the message stamped with current time is written to the client's sending buffer. In the sending buffer, acknowledged messages are removed while unacknowledged messages during a given time period are resent using random back-off timers to avoid retransmission flooding. Thus, lost messages to server can by design be recovered in SSTP, while lost messages to clients cannot be recovered. Reliable message delivery to clients can be implemented at the applications level. In the face of persistent server failures, clients can establish new secure connections with an alternative server.

From the latency perspective, a client can immediately send messages without any delay unless its sending window is full. In contrast, messages to clients are delayed to enable piggybacking on acknowledgements to clients. Each client's sending window size is determined by the number of the clients per server as the server expects its clients of evenly reducing their window size when it has to serve more than an expected number of clients.

IV. EVALUATIONS

A. Security and Overhead

We now recap the security of SSTP – much of the security argument appears along the presentation of the protocol design in Section III. First, the long-term key generation is secure due to the properties of PRFG. The follow-up AKE, chosen from the literature and standards, is also secure. Finally, the composition of the post-AKE secure session protocols with server offloading is secure if replay attacks are eliminated as shown in Section V.

From view point of computation and memory overhead, we compare SSTP to TLS over TCP (referred to as TLS hereafter). Consider two types of TLS client-server association: *long-lived* and *short-lived*. For the former, each client keeps an association with its server during the lifetime of secure transport connection. For the latter, it establishes a connection to its server to deliver a message and terminates the connection after message delivery. For a simple comparison, consider a large *N*-meters network where each meter (client) sends sensed data to its data collector (server) in a certain time intervalλ. Assume that the security and transport state per client consumes *L* memory space in a server.

Short-lived TLS has high computation overhead for control, since it repeatedly sets up a new connection per data every λ, processes *one* data item, and deletes the connection. S*eventeen* control messages are required to deliver the data (12 for client-authenticated TLS establishment [3] and 5 for TCP establishment and finish). It implies that short-lived TLS servers severely use CPU, memory, and network resources just for processing control messages, if λ is small. On the other hand, long-lived TLS servers need O(*NL*) memory space to maintain security-transport state for clients and also extra computation overhead to lookup one corresponding state across O(*N*) state. It means that long-lived TLS servers excessively uses memory and also consumes CPU resource for the state lookup if λ is large. Comparatively, SSTP servers have no overhead in computation or memory usage as observed in alternatives irrespective of λ and *N* because of stateless design. The overhead shown in SSTP is state-token processing done per data received, which is not higher than the state lookup of long-lived TLS. From the communication aspect, SSTP clearly outperforms short-lived TLS exchanging control messages every λ, and is similar to long-lived TLS since the header size of the SSTP is comparable to that of TLS. We stress that in SSTP, clients do not by design send connection requests in the face of temporal server failures, compared to alternatives where clients develop SYN flooding by sending the requests within a time boundary.

B. Network Delay

We consider a congestion-free network expected in smart grid data collection. However, this assumption is less favorable for SSTP; in a congested network we expect the performance of SSTP to be further improved relative to TLS. In the future, we plan to test the performance of SSTP under network congestion.

Simulation Setting: We measure the end-to-end delay of TCP Reno and SSTP under ns-2 simulation settings summarized in Table II. A data source, representing a sensing node, continuously injects 512byte packets into the network in a periodic interval. As HSPA or WiMAX are expected to be used for smart-grid access network technology, results [19] of speedtest in Chicago Oct. 2010 are used for this simulation setting: topology 1 represents ATT HSPA and topology 2 represents Clearwire WiMAX. TCP Reno is compared against SSTP because [20] showed that TCP Cubic (the “better” TCP variant) is outperformed by TCP Reno in terms of delay. Note that this evaluation assumes that there is no loss, drop, and reordering of messages. Thus this simulation focuses on the effect of network delay and is free of the adverse consequences of congestion control schemes and in-order sequence service.

Table II: NS-2 SIMULATION SETTINGS.

Network settings	TCP Reno vs SSTP	
- Uplink Speed - Uplink Delay - Downlink Speed - Downlink Delay	278 kbps [topology 1] 240 ms 2.8 Mbps 160 ms	1 Mbps [topology 2] 120 ms 6 Mbps 80ms
TCP Specific settings	536B MSS, 64KB RX Buffer, and 40ms Delayed Ack.	
Sensing Data Size	512 Byte Data over uplinks, ACK over downlinks	
Data Injection Speed	10ms ~250ms (441.6kbps ~ 1.76kbps)	
Simulation Duration	3 minutes after steady state	

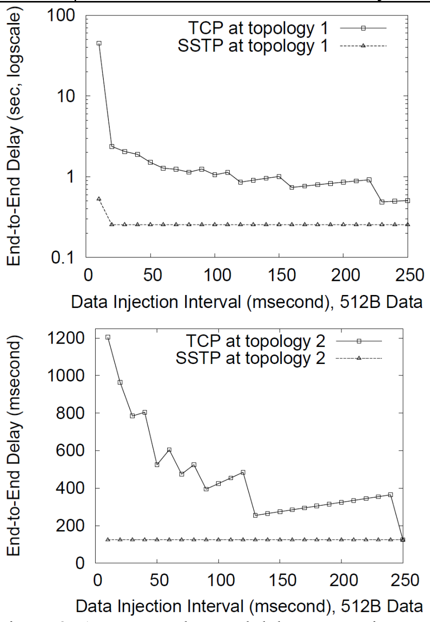


Figure 3: Average end-to-end delay comparisons.

Simulation results: As expected**,** SSTP outperforms TCP across the topologies described in Table II. Figure 3 represents average end-to-end delays of SSTP and TCP as a function of constant bit-rate workloads. SSTP keeps pace with the physical link speed under all settings except 10 ms data injection interval and 278 kbps bandwidth. In contrast, TCP always shows larger end-to-end delay than RTT under conditions where the data injection interval is smaller than RTT. In this evaluation, we notice two properties of TCP. First, TCP’s end-to-end delay is extremely high, about 46 sec when the network bandwidth is not sufficient for transmission, as shown in Figure 3. Second, with enough bandwidth, network delay dominates TCP’s end-to-end delay. Interestingly the simulation result on topology 2 shows that TCP is comparable to SSTP in large data injection intervals.

Based on our observation, we see that TCP works well only under the following condition: Before the size of the unacknowledged sending window reaches a congestion window limit, the ACK is always returned to the sender and then data is pushed from an application. Namely, when the data injection interval is greater than RTT, and data is contained in one TCP

MSS, TCP's end-to-end delay is close to one-way time from a sender to a receiver, due to the always-open congestion window. Otherwise, TCP shows the sensitivity to both data injection interval and RTT (varies on media and routing between communicating pairs). For example, under the condition where the data injection interval is greater than round trip timeout (RTO) value (minute scale), TCP's end-to-end delay is at least RTT without packet drop, if the packet size is more than one TCP MSS, and at least the RTO value, even on single a packet drop, irrespective of the packet size. In contrast, in terms of delay, SSTP works well and is independent of the data injection interval and network settings.

V. Discussion on SSTP Vulnerabilities

The SSTP server is essentially stateless and may be vulnerable to *replay* attacks. We now discuss how we protect the server from this type of attack. First, we note that server's secret-key-based authentications of the state-tokens are un-forgeable and with proper formatting and care one can prevent the adversary from presenting a state-token generated for a client id_1 as a state-token for client id_2. Further we will use deterministic encryption, such as AES, so re-encrypting a state-token without knowledge of a secret key is not possible either. Hence, the only venue of the replay attack is the verbatim replay of one of the previously generated state tokens with a possibly different session message. We observe that session message is implicitly tied with the state-token, since the session message (due to the necessary replay protection inside secure session protocols) is cryptographically tied with the client's and server's states, and thus with the state-token. Thus, the only replay attack that remains to be considered is the verbatim replay of the entire client's message. And indeed, our presentation so far is vulnerable to this attack. We now discuss our protection technique. Firstly, a server will reject "obviously old" messages through checking the token issue time contained in a state-token. Still, we need to efficiently address the possibility of replay of "not obviously too old" messages, which might be up to several tens of seconds old (to allow for clock skew). Our observation is that the number of messages that can arrive in this time period of tens of seconds is not very large, and therefore we can afford to keep the history of their hashes. For each new message, we will check it against the small recent history of hashes, and reject it if it is found in the history; if not found, we proceed as before. One optimization that we can implement is the use of Bloom filters [21] to greatly reduce the hash table size, and to speed up the hash checks.

VI. Conclusion

In this work, we show that existing known protocols do not meet the scalable secure transport requirements for smart grid sensor data collection. Motivated by this, we design *SSTP,* which achieves inherent security and meets transport requirements. Our evaluation confirms that a combination of TCP with TLS has scalability issues for sensor data collection in large-scale networks, and TCP by itself is inappropriate for periodic sensor data collection. For further study, we will consider SSTP enhanced with a TCP-friendly congestion control scheme [22] for sensor data collection over public networks.

VII. Related Work

Relevant to our work is the body of recent research that has examined end-to-end transport solutions over power line communications [23] and RF mesh network [24]. We can conclude from this literature that existing transport solutions are not suitable for smart grid sensor data collection, which relies on millisecond to minute scale measurements: average delay between PLC slaves (meters) and a PLC master is about 10 minutes even in a 100-nodes network [23]. Up to one hour delay was observed in a RF mesh network [24]. More importantly, these prior solutions do not address scalable end-to-end security extensions. Alternative directions are split-aggregation concepts [10], where high-powered intermediate nodes aggregate data and effectively respond to congestions through a hop-by-hop delivery scheme, and so retransmissions are reduced. Under the condition where congestion is rare, the impact of these alternatives is confined to RTT reduction. However, the main concern with this approach is that the end-to-end security can be hurt since the split-aggregation requires TLS or IPSec sessions to be terminated at intermediate nodes. We stress that SSTP can be directly applied to the split-aggregation model after minor modifications (omitted due to space constraints).

References

[1] "Assessment of Demand Response and Advance Metering," Federal Energy Regulatory Commission, Staff Report Docket No.: AD-06-2-000, Aug., 2006.
[2] "TI MSP430 product brochure", http://focus.ti.com/lit/sg/slab034t.pdf
[3] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol," IETF RFC 5246, Aug., 2008.
[4] S.Kent and K. Seo, "Security Architecture for the Internet Protocol", IETF RFC 4301, Dec., 2005.
[5] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," IETF RFC 4987, Aug., 2007.
[6] P. Metzger, W. Simpson, and P. Vixie, "Improving TCP Security With Robust Cookies," USENIX Magazine, Vol.34, No.6, Dec., 2009.
[7] R. Stewart, "Stream Control Transmission Protocol", RFC4960, Sep., 2007.
[8] K. Ono and H. Schulzrine, "The Impact of SCTP on SIP Server Scalability and Performance," IEEE GLOBECOM, Nov., 2008.
[9] "Reduce Memory Footprint of TCP/IP stack", http://infocenter.arm.com /help/index.jsp?topic=/com.arm.doc.faqs/ka11285.html
[10] T. Khalifa, K. Naik, M. Alsabaan, A. Nayak, and N. Goel, "Transport Protocol for Smart Grid Infrastructure", IEEE UFN, June, 2010.
[11] R. Braden, "Requirements for Internet Hosts - Communication Layers," RFC1122, Oct., 1989.
[12] D. Bakken, C. Hauser, and H. Gjermundrod, "Delivery Requirements and Implementation Guidelines for the NASPI net Data Bus," Proc. of IEEE SmartGridComm, Oct., 2010.
[13] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," IEEE Infocom, Mar. 2000.
[14] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control", IETF RFC5681, Sep., 2009.
[15] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective ACK Options," IETF RFC 2018, Oct. 1996.
[16] R. Stewart, M. Tuexen, and I. Ruengeler, "SCTP NAT," draft-stewart-behave-sctp-nat-04.txt, July., 2008.
[17] E. Rescorla, "Diffie-Hellman Key Agreement Method," IETF RFC 2631, June, 1999.
[18] P. Eronen and H. Tschofenig,"Pre-Shared Key Cipher suites for Transport Layer Security (TLS),", IETF RFC4279,Dec., 2005.
[19] "Review: Network speed test –WiMAX vs 3G," Oct, 2010, http://www.rcrwireless.com/ARTICLE/20101026/WIRELESS_FACTS_AND _FIGURES/101029953/review-network-speed-test-8211-wimax-vs-3g.
[20] E. Halepovic, Q. Wu, C. Williamson, and M. Ghaderi, "TCP over WiMAX: A Measurement Study," IEEE MASCTOS, Sep., 2008.
[21] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, Vol.13, No.7, Jul., 1970.
[22] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol," IETF RFC 4340, Mar., 2006.
[23] M. Bauer, W. Plappert, C. Wang, and K. Dostert, "Packet-Oriented Communication Protocols for Smart Grid Services over Low-Speed PLC," IEEE PLC and Its Applications, Mar., 2009.
[24] J. Paek and R. Govindan, "Rate-Controlled Reliable Transport Protocol for Wireless Sensor Networks," ACM SENSYS, Nov. 2007.