Multicast Authentication in Smart Grid With One-Time Signature

Qinghua Li, *Student Member, IEEE*, Guohong Cao, *Fellow, IEEE*

***Abstract*— Multicast has been envisioned to be useful in many Smart Grid applications such as demand-response, wide area protection, in-substation protection, and various operation and control. Since the multicast messages are related to critical control, authentication is necessary to prevent message forgery attacks. In this paper, we first identify the requirements of multicast communication and multicast authentication in Smart Grid. Based on these requirements, we find that one-time signature based multicast authentication is a promising solution, due to its short authentication delay and low computation cost. However, existing one-time signatures are not designed for Smart Grid, and they may have high storage and bandwidth overhead. To address this problem, we propose a new one-time signature scheme which can reduce the storage cost by a factor of 8 and reduce the signature size by 40% compared with existing schemes. Thus, our scheme is more appropriate for Smart Grid applications where the receivers have limited storage (e.g., home appliances and field devices) or where data communication is frequent and short (e.g., phasor data). These gains are at the cost of increased computations in signature generation and/or verification, and fortunately our scheme can flexibly allocate the computations between the sender and receiver based on their computing resources. We formulate the computation allocation as a nonlinear integer programming problem to minimize the signing cost under a certain verification cost, and propose a heuristic solution to solve it.**

Index Terms—**Smart Grid**, **Security**, **Multicast**, **Authentication**, **One-Time Signature**.

I. Introduction

Power grid is a critical infrastructure which, if disrupted or destroyed, has serious impacts on the safety of citizens, the stability of the economy, and the effective functioning of the whole society. The current power grid is evolving toward a more efficient and reliable *Smart Grid*, which will be featured by renewable-based clean generation, distributed micro-generators, wide area monitoring and control, smart metering, etc. The core of Smart Grid is an intelligent communication system that links all components together in an efficient and secure manner, and enables the two-way flow of electricity and information between the utility and the consumers, and all points in between.

Multicast enables one-to-many communications in an efficient way. Although multicast has been widely studied in the Internet and more recently in wireless sensor networks [27] and disruption tolerant networks [7], its application in critical infrastructures like Smart Grid has not received much attention. In Smart Grid, multicast has many applications. For example, in wide area protection, Phasor Measurement Units (PMUs) can be used to measure system parameters such as voltage and current, and then multicast the data to control centers. Based on the received data, control centers take appropriate actions to prevent cascaded failures [6]. As another example, during periods of peak energy consumption, utility centers can multicast a demand-response command to a large group of home appliances asking them to temporarily turn off or shift to a lower power level. In addition, many other applications use multicast for operation and control in Smart Grid.

Considering multicast messages are either measurement data or control commands, it is important to authenticate these messages so that each receiver can verify if the received message comes from the claimed sender and has not been tempered during the transmission. Without authentication, an attacker can easily modify an in-transit message, forge an arbitrary message, or replay an old message to trigger false and even catastrophic actions. The forgery of control command has been successfully exploited to attack a water utility [25], and similar attacks may be launched to Smart Grid.

Despite its importance, multicast authentication in Smart Grid is still an open problem due to the unique requirements of Smart Grid. Since most multicast messages are time-critical and field devices usually have limited resources, authentication should be done quickly and efficiently. Thus, traditional public key based digital signatures like RSA are too computation-intensive to be applied. Also, hybrid approaches [21], [26], [15] that amortize the public-key signature over multiple messages and delayed key disclosure based schemes [21], [14] cannot be applied since they have a significant authentication delay caused by message buffering. Several recent works [20], [19], [28] rely on One-Time Signature (OTS) such as BiBa [20] and HORS [24] to provide instant authentication for multicast messages. Constructed upon one-way functions without trapdoors, OTS is very efficient in computation. However, the storage overhead of existing OTS schemes is still too high for Smart Grid applications where the receivers such as home appliances and field devices, have limited storage. Also, existing OTS schemes have large signature size which can increase the bandwidth requirement.

In this paper, we propose a new OTS scheme to address the aforementioned problems. Compared with HORS, our scheme reduces the storage overhead of the receivers by a factor of 8, which is very important when receivers are resource-constrained. Also, our scheme reduces the signature size by 40%, which means a significant reduction in communication bandwidth for applications that require a high multicast frequency and a small message size (e.g., phasor data). Though our scheme increases the computation overhead for signature generation and/or verification, it can flexibly allocate the

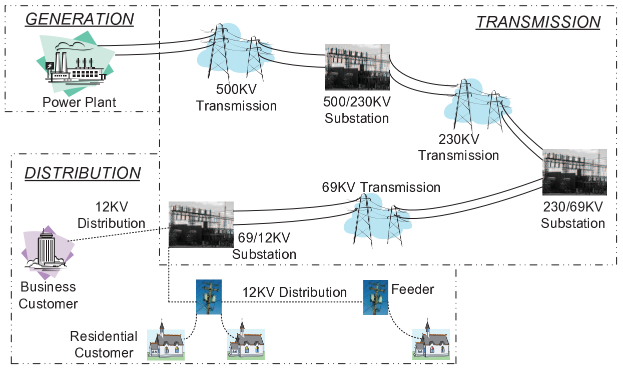


Fig. 1. The concept of power generation, transmission and distribution.

increased computations between the sender and receiver based on their computing resources. We formulate the computation allocation as a nonlinear integer programming problem to minimize the signing cost under a certain verification cost and propose a heuristic solution to solve it. Also, we design a multicast authentication protocol based on the proposed OTS scheme.

This paper is structured as follows. Section II introduces multicast communications in Smart Grid, analyzes the requirements on multicast authentication, and reviews related work. Section III describes our OTS scheme. Section IV formulates the computation allocation problem and presents a heuristic solution. Section V presents the multicast authentication protocol. Section VI evaluates the proposed solution, and Section VII concludes the paper.

II. Multicast in Smart Grid

In this section, we first introduce how the power grid works, and then present several multicast communication applications in Smart Grid. We also discuss the security threats to multicast, the requirements of multicast authentication, and related work in this area.

A. Power Grid Overview

Figure 1 shows the concept of a typical power generation, transmission, and distribution system. The power plant generates electricity and increases the voltage to a very high level. The high-voltage electricity is transmitted to a distant *transmission substation* where the voltage is reduced and the electricity is further transmitted to lower-level transmission substations. Finally the electricity is transmitted to local *distribution substations* which will distribute it to consumers. Several distribution lines emanate from each distribution substation and each line supplies a number of consumers such as residential houses and other local loads.

The safe operation of power grid requires that critical electricity parameters (e.g., voltage and frequency) always stay within their operating ranges. However, unexpected events like short circuits and the imbalance between power demand and supply may affect these electricity parameters which may be out of their operating ranges. For safe operation, those parameters are monitored and controlled in realtime. In some extreme cases, protection schemes are triggered to separate a problematic grid component.

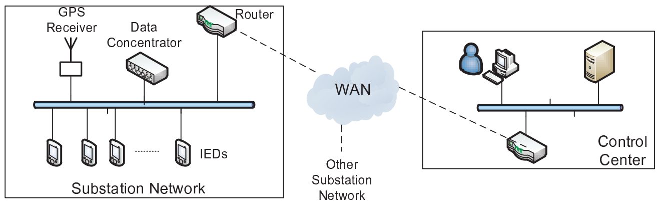


Fig. 2. The typical architecture of a SCADA system. IEDs refer to Intelligent Electronic Devices equipped with communication capabilities.

Smart Grid will bring new features into power grid such as renewable-based generation, demand-response, wide area protection, smart metering, etc. The core of Smart Grid is an intelligent communication system that links all components together in an efficient and secure manner. Smart grid communications can be unicast or multicast. Unicast has many applications such as sending a measurement report from a field device to the control center, and sending a control command in the reverse direction. Multicast also plays an important role, and will be the focus of this paper.

B. Multicast Applications

1) Wide Area Protection: When the grid topology suddenly changes due to unexpected loss of a large generator, transmission line or load, protective actions may be triggered to disconnect other generators or transmission lines, resulting in a cascaded failure. Such cascaded failures caused the well known Northeast Blackout in 2003 [6].

Wide area protection schemes are being deployed to prevent cascaded failures. In these schemes, Phasor Measurement Units (PMUs) measure system parameters such as current and voltage at precisely synchronized times and multicast the phasor data to the control centers. For example, in the North American SynchroPhasor Initiative [8], each PMU multicasts measured system parameters to the control centers at a rate of 30 times per second. From the phasor data, the control centers detect problems like electricity frequency drop, and issue control commands to open or close appropriate switches.

2) Demand-Response: During periods of peak energy consumption, utility companies send alerts to ask consumers to reduce their power consumption by temporarily turning off non-essential appliances. If enough consumers comply with the requests, the power reduction could be enough to avoid building an additional expensive power plant. With support from variable-rate pricing and smart metering technologies that measure energy usage at different time of a day, demand-response is expected to be widely deployed in Smart Grid [10]. Since millions of appliances may be involved, unicast may not be a good solution to transmit the alert message. Thus, for demand-response, the control center generally multicasts alert messages to a large number of remote appliances.

3) Operation and Control: For safe operation, some control actions such as opening a circuit switch will be sent from the control center to remote field devices. The control commands are delivered through the Supervisory Control and Data Acquisition (SCADA) system (see Figure 2). In many cases, a control command should be sent to a large number of field devices and it should be executed immediately, and

TABLE I

A SUMMARY OF MULTICAST COMMUNICATIONS IN SMART GRID

Category	Allowable Delay	Frequency	Sender	Sender Resource	Receiver	Receiver Resource
Wide Area Protection	a few tens of ms	high (30/s)	PMU	limited or moderate	control center	sufficient
Demand-Response	a few seconds	low	control center	sufficient	home appliance	limited
Operation and Control	tens of or a hundred of ms	low	control center	sufficient	field device	limited or moderate
In-Substation Protection	a few ms	low	protective relay	limited or moderate	circuit breaker	limited

TABLE II

REQUIREMENTS OF MULTICAST AUTHENTICATION ON COMPUTATION, STORAGE, AND COMMUNICATION COMPLEXITY

Category	Sender Computation	Receiver Computation	Sender Storage	Receiver Storage	Bandwidth
Wide Area Protection	stringent	loose	less stringent	loose	stringent
Demand-Response	loose	less stringent	loose	stringent	loose
Operation and Control	loose	stringent	loose	stringent	loose
In-Substation Protection	stringent	stringent	less stringent	stringent	loose

hence multicast is the right choice. For example, in case of emergency, the control center multicasts important messages such as “emergency shutdown” to all or a large fraction of substations and field devices. When the power grid cannot supply all consumer loads, it must disconnect some consumers. A straightforward approach is to disconnect a high-load transmission line, but it will probably result in a large-area blackout and huge revenue loss. In Smart Grid, *fine-grained* load shedding should be adopted. Instead of disconnecting the whole area, only a number of less-important consumers are disconnected. In this scenario, multicast is needed to transmit the disconnect command to the relevant distribution feeders and smart meters.

4) In-Substation Protection: In a substation, multicast can be used to disseminate time-critical messages like fault alerts across substation LANs. When detecting a fault, the protective relay multicasts a command to the appropriate circuit breakers to disconnect the faulty circuits. For this purpose, a link-layer multicast protocol is designed in IEC61850 [1]. Usually, the message must be transmitted within a few milliseconds, e.g., 4 ms in an IEEE standard [9].

Table I summarizes the four classes of multicasts used in Smart Grid.

C. Security Threats

An adversary (such as a terrorist or a disgruntled ex-employee) may launch cyber security attacks to the power grid by forging multicast messages. To do this, the adversary can eavesdrop the communication channel and intercept a signed message that the sender multicasts to receivers. From the information in the intercepted message, the adversary can forge a signature for her own message, and then inject her own message into the communication channel, which will be multicast to the receivers. Note that the adversary can also compromise a receiver to get a valid signed message instead of eavesdropping the communication channel. With the forged message and signature, the adversary can cause great damage to the power grid. For example, when the receivers are the distribution feeders that supply consumers in a large area, the adversary can include a disconnect command in the injected message which will cause a large-area breakout. Thus, it is important to authenticate multicast messages so that the receiver can verify if the messages come from the claimed sender and have not been modified during the transmission.

D. Requirements on Multicast Authentication

Different applications in Smart Grid have different requirements on multicast authentication in terms of computation, communication, and storage complexity, as summarized in Table II.

The bandwidth cost of authentication should be as small as possible in wide area protection, since phasor data messages are transmitted at a very high frequency. However, bandwidth is not a big concern for the other three applications. Also, the storage cost at the receiver side should be kept low when the receivers are home appliances or field devices with very limited storage. Since the delay requirement is stringent in wide area protection, operation and control, and in-substation protection, the computation cost of authentication should be low for the devices with constrained computing resources. In demand-response, because the time requirement is less stringent the computation overhead at the receiver side is less important. Generally speaking, the computation burden on the control center is not a big concern, but it cannot be too high.

E. Literature Review

The most straightforward solution to multicast authentication is to use public key cryptography (PKC) based digital signatures like RSA. However, these signatures have too much computation cost for Smart Grid since most field devices and home appliances are resource-constrained and they may not be able to sign or verify a message within the time constraint.

Hybrid approaches [21], [26], [15], [22], [11] have been proposed to reduce the computation cost. These approaches combine PKC with efficient one-way functions. Instead of generating one digital signature for each message, they generate one signature for multiple messages, and the verification of the signature authenticates all those messages. In this way, the cost of PKC-based signing and verification is amortized over multiple messages. In these approaches, however, a message (before being authenticated) must be buffered by the sender or receiver until the last message that shares the same signature is available. Hence, the authentication delay may be long, and cannot meet the time requirement of Smart Grid applications.

Perrig *et al* [21] proposed TESLA, a multicast authentication protocol that completely relies on symmetric key cryptography. TESLA is based on the delayed disclosure of authentication keys, i.e., the key used to authentication a message is disclosed in the next message. Several later works [13], [14], [4] improve the performance of TESLA with techniques such as multi-level one-way chains and Merkle hash trees. However, in TESLA-based protocols, a message has to be buffered for some time at the sender or receiver. Thus, they are not appropriate for Smart Grid due to the delay.

One Time Signature: OTS is a promising solution for multicast authentication in Smart Grid, since it can provide instantaneous authentication without message buffering delay and it can tolerate the compromise of some receiving nodes. OTS is conceptually similar to PKC-based signatures in that the sender uses a private key to sign a message and the receiver uses a public key to verify the signature. However, OTS is much more efficient in computation since it is built upon one-way functions without trapdoors.

OTS was proposed independently by Lamport [12] and by Rabin [23], and then improved by several works [16], [17], [2]. In these schemes the signature size can be hundreds and even thousands of bytes, which is too large. Recently, Perrig [20] proposed the BiBa signature which reduces the signature size to 130 bytes. The disadvantage of BiBa is that it requires a long time to sign a message. To reduce the signing cost, Reyzin and Reyzin [24] proposed the HORS signature. HORS only needs one hash computation to sign a message, making it the fastest OTS in signature generation. HORS also improves the signature verification cost, and its signature size is similar to that of BiBa. Due to its efficiency, HORS has been used in several works [19], [28] to authenticate time-critical multicast messages.

However, HORS has some weaknesses when applied to Smart Grid. First, in applications where the receivers are resource constrained, the public key size of HORS is too large, which means a high storage overhead at the receiver side. Though some recent work [3] improves the public key size of HORS, the scheme cannot be applied to one-way chain based authentication protocols [20], [19], [28], and thus the distribution of the public key becomes an issue. Second, the signature size of HORS is too large for the wide area protection application. In a typical setting, one HORS signature has 130 bytes, but a phasor data frame may only have 48 bytes based on the IEEE C37.118 standard.

III. Our OTS Scheme

In this section, we propose a new OTS scheme that addresses the weaknesses of HORS. We first review HORS to give some background and then present our OTS scheme.

A. HORS [24]

HORS uses a cryptographically strong hash function H to map each message (m) to a k-element subset of a t-element set (T). The private key is T, the public key is the set created by applying a one-way function to each element of T, and the signature is the k-element subset that m is mapped to. The protocol is as follows:

- **Key Generation** Generate t random l-bit strings $(s_1, s_2, ..., s_t)$, which form the private key SK. The public key is then computed as $PK = (v_1, v_2, ..., v_t)$, where $v_i = f(s_i)$ and f is a one-way function.
- **Signing** To sign a message m, let $h = H(m)$, where H is a hash function. Split h into k substrings $h_1, h_2, ..., h_k$ of $\log_2 t$ bits each. Interpret each h_j as an integer i_j. The signature of m is $(s_{i_1}, s_{i_2}, ..., s_{i_k})$.
- **Verification** To verify a signature $(s'_1, s'_2, ..., s'_k)$ over the message m, compute $h = H(m)$. Split h into k substrings $h_1, h_2, ..., h_k$ of $\log_2 t$ bits each. Interpret each h_j as an integer i_j and check if $f(s'_j) = v_{i_j}$ holds.

B. Our Basic Idea

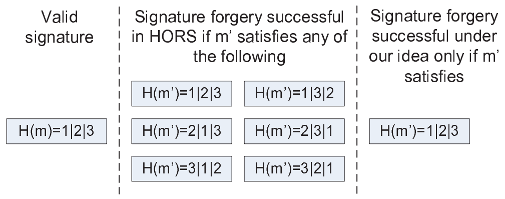

Valid signature
Signature forgery successful in HORS if m' satisfies any of the following
Signature forgery successful under our idea only if m' satisfies
H(m)=1|2|3
H(m')=1|2|3
H(m')=1|3|2
H(m')=2|1|3
H(m')=2|3|1
H(m')=3|1|2
H(m')=3|2|1
H(m')=1|2|3

Fig. 3. An example of signature forgery in HORS and in our approach. Suppose $k = 3$. The adversary has a valid signature for message m, and it wants to forge a signature for m'.

In HORS, the k elements of a signature are verified in the same way. Suppose an adversary has eavesdropped a valid signature $(s_{i_1}, s_{i_2}, ..., s_{i_k})$. Then it can forge a signature for its own message m' if in the signing step it can map m' to *any* of the $k!$ permutations of $s_{i_1}, s_{i_2}, ..., s_{i_k}$, i.e., the k substrings of $H(m')$ when interpreted as integers form any permutation of $i_1, i_2, ..., i_k$. An example is shown in Figure 3.

To make signature forgery more difficult, our idea is to consider the k elements of a signature as an ordered sequence pattern in which the position of each element can determine the signature verification process. Then, from a valid signature $(s_{i_1}, s_{i_2}, ..., s_{i_k})$, the adversary can only obtain one sequence pattern as included in the signature. To forge a signature, the adversary must map its own message to this exact sequence, as shown in Fig. 3. Thus, with the same parameters, the difficulty (or computational complexity) of signature forgery is increased by a factor of $k!$ compared with that in HORS. (Suppose $i_1, i_2, ..., i_k$ are different.) For example, when $k = 13$ as in a typical setting of HORS where $t = 1024$ and the security level is 80-bit, the factor is as large as 2^{28}.

In HORS the computational complexity of signature forgery, given by $(\frac{t}{k})^k$, is positively related to t and k. Therefore, with our idea, we can achieve the same security level with a smaller public key size t. This will result in a smaller storage cost at the receiver side, which is meaningful in multicast applications such as demand-response, operation and control, and in-substation protection, where receivers have limited storage. Alternatively, the same security can be achieved with a smaller signature size k, i.e., a smaller communication cost. This is useful for multicast applications such as wide area protection where communication bandwidth is important.

The benefit on storage and bandwidth is at the cost of increased computations at the sender or receiver. Fortunately, by implementing the idea in an adaptive way, the computations can be flexibly allocated to the sender or receiver based on their computing resources. This allows us to trade non-critical computation resources for savings in critical resources. For example, in demand-response it is reasonable to increase the computation cost of sender (i.e., control center) to reduce the storage cost of receivers (i.e., home appliances).

In our scheme, $s_1, s_2, ..., s_t$ are t different random l-bit strings, which form the seeds of the private key. f denotes a one-way function without trapdoors. $f^x(s)$ is the result of applying f over s for x times. $f^0(s) = s$ and $f^1 s = f(s)$. The *index* of signature element s is defined as the integer i that satisfies $s = f^x(s_i)$ with some $x \geq 0$.

Before presenting our scheme, we define a function *SPLIT*(h) which takes bit string h as input and outputs k integers $i_1, i_2, ..., i_k$. This function splits h into k substrings $h_1, h_2, ..., h_k$ of $\log_2 t$ bits each and interprets each h_j as an integer i_j.

C. Two Extremes

Before proceeding to our scheme, we first present two extreme schemes which use two distinct approaches to implement our basic idea.

1) Heavy Signing Light Verification (HSLV): HSLV mainly changes the signing process of HORS, and it obtains better security with higher signing overhead. Specifically, it requires that the k elements of a signature are sorted in the decreasing order of their index. The detailed protocol is as follows:

- **Key Generation** The same as that of HORS (see Section III-A).
- **Signing** To sign a message m, compute $h = H(m|c)$, where c is a counter with initial value 0. Call *SPLIT*(h). All i_j from *SPLIT*(h) should be different and satisfy $i_1 > i_2 > ... > i_k$; otherwise, increase c by 1 and repeat the above process. The signature is $(c, (s_{i_1}, s_{i_2}, ..., s_{i_k}))$.
- **Verification** To verify a signature $(c', (s'_1, s'_2, ..., s'_k))$ over message m, compute $h = H(m|c')$. Call *SPLIT*(h). Check if the output integers $i_1 > i_2 > ... > i_k$ and $f(s'_j) = v_{i_j}$ hold.

Suppose the hash function H generates random bit strings. Then on average the sender (i.e., signer) needs to invoke H for at least $k!$ times to map a message to k sorted integers. When k is not very small, this signing cost may be too high even for a powerful computer.

2) Light Signing Heavy Verification (LSHV): LSHV mainly changes the signature verification process of HORS, and it obtains better security with higher verification overhead. Specifically, it verifies the k elements of a signature by applying the one-way function for a distinct number of times over each element. The detailed protocol is as follows:

- **Key Generation** Generate t different random l-bit strings $(s_1, s_2, ..., s_t)$. For each s_i, generate a one-way chain of length k, i.e., $s_i \rightarrow f(s_i) \rightarrow ... \rightarrow f^{k-1}(s_i)$. The t chains form the private key SK, and the public key is $PK = (v_1, v_2, ..., v_t)$, where $v_i = f^k(s_i)$ $(1 \leq i \leq t)$.
- **Signing** To sign a message m, compute $h = H(m|c)$, where c is a counter with initial value 0. Call *SPLIT*(h). All i_j from *SPLIT*(h) should be different; otherwise, increase c by 1 and repeat the above process. The signature is $(c, (f^{k-1}(s_{i_1}), f^{k-2}(s_{i_2}), ..., f^0(s_{i_k})))$.
- **Verification** To verify a signature $(c', (s'_1, s'_2, ..., s'_k))$ over the message m, compute $h = H(m|c')$. Call *SPLIT*(h). Check if $f^j(s'_j) = v_{i_j}$ holds.

The sender may apply H multiple times over a message to make sure all substrings of the hash result are different.

In LSHV, the signing cost is very low but the verification cost is higher. The comparison between LSHV and HSLV indicates that the receiver can take more computations to reduce the signing cost. This motivates us to combine LSHV with HSLV in an integrated way that leads to our scheme.

D. Our Scheme

We combine HSLV with LSHV and obtain the scheme Tunable Signing and Verification (TSV) which achieves a flexible tradeoff between the two. TSV divides the k elements of a signature into a number of groups according to their position in the signature. The elements in the same group are verified with the same number of one-way function invocations, but this number is distinct for each group. Also, the elements in the same group are sorted in the decreasing order of their index. Thus, elements in the same group are processed similarly as in HSLV, while elements in different groups are processed similarly as in LSHV.

Suppose the k elements of a signature are divided into g groups $\mathbb{G}_1$, $\mathbb{G}_2$, ..., $\mathbb{G}_g$. Let n_r $(r \in [1, g])$ denote the size of group $\mathbb{G}_r$. Then $\mathbb{G}_1$ contains the first n_1 elements, $\mathbb{G}_2$ contains the next n_2 elements, etc. Let $\mathbb{G}_{q_i}$ denote the group to which the i^{th} $(i \in [1, k])$ element in the signature belongs. Each element in group $\mathbb{G}_r$ is verified with $w_r + 1$ $(w_r \geq 0)$ one-way function invocations. Since at least one invocation is needed for each element and this part of cost is fixed, w_r measures the flexible part of the verification cost. Let $w = max\{w_1, ..., w_g\}$. Then the protocol is as follows:

- **Key Generation** Generate t different random l-bit strings $(s_1, s_2, ..., s_t)$. For each s_i, generate a one-way chain of length $w + 1$, i.e., $s_i \rightarrow f(s_i) \rightarrow ... \rightarrow f^w(s_i)$. The t chains form the private key SK. The public key is $PK = (v_1, v_2, ..., v_t)$, where $v_i = f^{w+1}(s_i)$.
- **Signing** To sign a message m, compute $h = H(m|c)$, where c is a counter with initial value 0. Call *SPLIT*(h). All i_j from *SPLIT*(h) should be different and the i_j within the same group[1] should be sorted in the decreasing order; otherwise, increase c by 1 and repeat the above process. The signature of m is $(c, f^{w-w_{q_1}}(s_{i_1}), ..., f^{w-w_{q_k}}(s_{i_k}))$.
- **Verification** To verify a signature $(c', (s'_1, s'_2, ..., s'_k))$ over the message m, compute $h = H(m|c')$. Call *SPLIT*(h). Check if 1) all i_j from *SPLIT*(h) are different, 2) the i_j in the same group are sorted in the decreasing order, and 3) $f^{w_{q_j}+1}(s'_j) = v_{i_j}$ for each j.

[1]Here we say integer i_j and $i_{j'}$ are in the same group if signature element s_{i_j} and $s_{i_{j'}}$ belong to the same group.

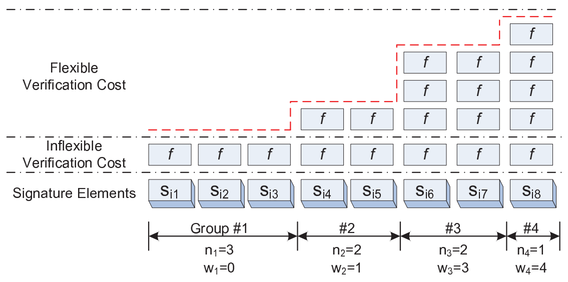

Flexible Verification Cost
Inflexible Verification Cost
Signature Elements
f
S_{i1} S_{i2} S_{i3} S_{i4} S_{i5} S_{i6} S_{i7} S_{i8}
Group #1 $n_1=3$ $w_1=0$
#2 $n_2=2$ $w_2=1$
#3 $n_3=2$ $w_3=3$
#4 $n_4=1$ $w_4=4$

Fig. 4. An example configuration TSV$(4, [3, 2, 2, 1], [0, 1, 3, 4])$ when each signature has 8 elements. A rectangle with label f means one invocation of one-way function f. The number of rectangles on top of each signature element represents the number of times f is applied to the element in verification.

TSV is configured by two vectors $\mathbf{n} = [n_1, ..., n_g]$ and $\mathbf{w} = [w_1, ..., w_g]$. With these notations, a TSV scheme can be represented by TSV$(g, \mathbf{n}, \mathbf{w})$. Note that HSLV = TSV$(1, [k], [0])$ and LSHV = TSV$(k, [1, 1, ..., 1], [0, 1, ..., k-1])$. An example configuration of TSV is also shown in Figure 4. The configuration of TSV is further addressed in Section IV.

E. Security Analysis

Assume an adversary has obtained a valid signature s'_1, ..., s'_k. We are interested in the probability P_f that the adversary is able to forge a signature for any message m' of its choice with one invocation of the hash function H. We assume the adversary cannot invert H or the one-way function f. We also assume H generates random bit strings.

In HSLV, the adversary must map m' to the exact sequence s'_1, ..., s'_k to pass the order check in verification. It also has to do so in LSHV, since each signature element is verified with a distinct number of one-way function invocations. Similar results hold for TSV. Now we consider the probability P_{f_j} that the adversary can map the j^{th} element of the forged signature to s'_j ($j \in [1, k]$). According to the signing process, P_{f_j} is equal to the probability that the integer i_j from the output of *SPLIT*$(H(m'))$ is the same as the index (see the definition in Section III-B) of s'_j. Since H generates random bit strings by our assumption, it is trivial to get $P_{f_j} = \frac{1}{t}$. Then $P_f = \prod_{j=1}^{k} P_{f_j} = (\frac{1}{t})^k$.

On average the adversary needs to perform $\frac{1}{P_f} = t^k$ hash invocations to forge a signature. This means that TSV achieves $\log t^k = k \log t$ bits of security, which is much better than that of HORS, i.e., $k(\log t - \log k)$ [24]. For instance, when $t = 1024$ and $k = 8$, the security level is 80-bit in TSV but only 56-bit in HORS.

F. Cost Analysis

We measure the computation cost of our scheme in terms of hash or one-way function invocations.

In TSV, the signer evaluates H for c times to find a qualified hash value. Let P_0 denote the probability that one hash invocation generates k different substrings, and P_r ($r \in [1, g]$) denote the probability that in one hash invocation the integers in group $\mathbb{G}_r$ are sorted in the decreasing order. Then the probability P_s that one hash invocation successfully generates a qualified signature is given by $P_s = \prod_{i=0}^{g} P_i$. Since H generates random bit strings, it is trivial to get $P_0 = \frac{t}{t}\frac{t-1}{t} \cdot \cdot \cdot \frac{t-k+1}{t} = \frac{t(t-1)\cdots(t-k+1)}{t^k}$. Also, in one invocation of H the integers in group $\mathbb{G}_r$ can appear as any of the $n_r!$ possible permutations with equal probability. Thus, $P_r = \frac{1}{n_r!}$. Then we get $P_s = \frac{t(t-1)\cdots(t-k+1)}{t^k} \prod_{r=1}^{g} \frac{1}{n_r!}$. On average the signing cost is $c = \frac{1}{P_s} = \frac{t^k}{t(t-1)\cdots(t-k+1)} \prod_{r=1}^{g}(n_r!)$.

The signature size is $kl + \log c$ bits. In typical settings where $t \gg k$, $\frac{t^k}{t(t-1)\cdots(t-k+1)}$ is close to 1 (e.g., 1.028 when $t = 1024$ and $k = 8$). Also, $\prod_{r=1}^{g}(n_r!) \leq k!$ as we will show in the next section. Furthermore, $k! \approx \sqrt{2\pi k}(\frac{k}{e})^k$ according to Stirling's approximation. Thus, we have $\log c \leq k \log k$. Since k is not very large and $l = 80$ [24], $k \log k$ is negligible compared with kl. Therefore, the signature size is approximately kl bits.

Table III lists the security and cost of TSV (including the special cases HSLV and LSHV) and compares it with other schemes.

IV. Optimizing the Computation Allocation

According to the analysis results in Table III, the signing and verification cost of TSV are determined by t, k, and the vector $\mathbf{n}$ and $\mathbf{w}$. Note that in reality t and k are selected based on factors such as the required security level and the available storage resource, and the selection is orthogonal to the following discussion. Thus, in this section we assume t and k are fixed without loss of generality. Then we focus on allocating the computation cost of TSV between signing and verification by adjusting the vector $\mathbf{n}$ and $\mathbf{w}$.

A. Problem Formulation

Let $\mathcal{C}$ denote the number of one-way function invocations that can be flexibly applied to any signature element, i.e., $\mathcal{C}$ is the flexible part of the verification cost. Obviously, $\mathcal{C} = \sum_{r=1}^{g} n_r w_r$. Given a certain $\mathcal{C}$, there exist many possible $\mathbf{n}$ and $\mathbf{w}$ with different signing cost. Then we consider the following computation allocation problem: *Given a certain $\mathcal{C}$, how to select $\mathbf{n}$ and $\mathbf{w}$ to minimize the singing cost?*

Since t and k are fixed, the first component of the signing cost $\mu = \frac{t^k}{t(t-1)\cdots(t-k+1)}$ is also fixed. In the following, we focus on the remaining components $\prod_{r=1}^{g}(n_r!)$ of the signing cost.

Obviously, when $\mathcal{C} = \frac{k(k-1)}{2}$ the signing cost can already reach its minimum if we set $\mathbf{n}$ and $\mathbf{w}$ as in LSHV. This means that it is meaningless to consider a larger value of $\mathcal{C}$. Also, when $\mathcal{C} = 0$, $\mathbf{n}$ and $\mathbf{w}$ can only be set as in HSLV, which is trivial to get. Thus, we only consider the cases $0 < \mathcal{C} < \frac{k(k-1)}{2}$.

In the verification process, the maximum possible number of flexible one-way function invocations applied to any signature element is $\mathcal{C}$. Thus, we can set $\mathbf{w} = [0, 1, ..., \mathcal{C}]$ and select the appropriate $\mathbf{n}$ to solve the computation allocation problem. We formulate the selection of $\mathbf{n}$ as the following problem:

$$min \prod_{r=0}^{\mathcal{C}}(n_r!) \quad s.t. \quad \sum_{r=0}^{\mathcal{C}} n_r r = \mathcal{C}, \sum_{r=0}^{\mathcal{C}} n_r = k, n_r \geq 0 \quad (1)$$

This is a nonlinear integer programming problem. Integer linear programming is well known to be NP-complete, and the nonlinearity further complicates the problem. Therefore, it is

TABLE III

THE SECURITY AND COST OF TSV AND OTHER OTS SCHEMES WITH THE SAME PARAMETER t AND k. LET $\mu = \frac{t^k}{t(t-1)\cdots(t-k+1)}$.

Scheme	Prob. of Forgery	Key Gen. Cost (offline)	Signing Cost	Verification Cost	Pub. Key Size	Sig. Size (bit)
BiBa [20]	$\geq \frac{k!}{2t^k}$	t	$2t$	$2k+1$	tl	kl
Powerball [18]	$\geq \frac{(k-1)!}{2t^k}$	$2t$	$2t$	$2k+1$	tl	kl
HORS [24]	$\frac{k^k}{t^k}$	t	1	k+1	tl	kl
HSLV	$\frac{1}{t^k}$	t	$k!\mu$	k+1	tl	$kl + \log{(k!\mu)}$
LSHV	$\frac{1}{t^k}$	kt	μ	$\frac{k(k+1)}{2}+1$	tl	$kl + \log \mu$
TSV($g, \mathbf{n}, \mathbf{w}$)	$\frac{1}{t^k}$	$(max\{w_1, ..., w_g\}+1)t$	$\mu \prod_{r=1}^{g}(n_r!)$	$1+k+\sum_{r=1}^{g} n_r w_r$	tl	$kl + \log{[\mu \prod_{r=1}^{g}(n_r!)]}$

difficult to find a general solution with polynomial time. Since the number of variables $\mathcal{C}$ can be large (e.g., 77 when $k=13$), an exhaustive search of the solution space is impractical. Thus, it is meaningful to reduce the solution space.

***Theorem** 1: There exists a solution to Formula 1 that satisfies* $\forall r \geq k, n_k = 0$.

Proof: See Appendix. ■

According to Theorem 1, the formulation in Formula 1 can be reduced to:

$$min \prod_{r=0}^{k-1}(n_r!) \quad s.t. \quad \sum_{r=0}^{k-1} n_r r = \mathcal{C}, \sum_{r=0}^{k-1} n_r = k, n_r \geq 0 \quad (2)$$

In this reduced formulation, the number of variables is k. When k is not large, an exhaustive search of the solution space becomes practical, especially considering that this problem can be solved off-line with powerful computers.

B. Heuristic Solution

Though Theorem 1 enables a significant reduction in the solution space, exhaustive search may still be impractical when k is large. Thus, we propose a heuristic algorithm.

We observe that when one additional one-way function invocation is applied to the last element in group $\mathbb{G}_r$, this element relocates to group $\mathbb{G}_{r+1}$. Let n_r and n_{r+1} denote the size of group $\mathbb{G}_r$ and $\mathbb{G}_{r+1}$ before relocation, respectively. Then it is easy to decrease the signing cost after relocation by a factor of $\alpha = \frac{n_r}{n_{r+1}+1}$. If $n_r > n_{r+1}+1$, then $\alpha > 1$ which means the signing cost decreases.

Based on this observation, our heuristic algorithm iteratively increases the flexible verification cost from 1 to $\mathcal{C}$, with an increment of one per step. In each step, the one-way function invocation is added to the signature element that results in the maximum signing cost reduction at this step, i.e., the last element of group $\mathbb{G}_r$ which satisfies $\forall r' \neq r$, $\frac{n_r}{n_{r+1}+1} \geq \frac{n_{r'}}{n_{r'+1}+1}$. If there is a tie, the smallest r is chosen. The algorithm is shown in Algorithm 1. The time complexity of this algorithm is $O(k\mathcal{C})$, i.e., $O(k^3)$ since $\mathcal{C} < \frac{k(k-1)}{2}$. The space complexity is $O(k)$. Thus, the algorithm is very efficient.

C. Tradeoff with Parameter $\mathcal{C}$

The observation in Section IV-B indicates that the signing cost decreases as $\mathcal{C}$ increases. According to the definition of $\mathcal{C}$, the verification cost increases as $\mathcal{C}$ increases. Thus, our scheme can achieve a tradeoff between the signing and verification cost by adjusting the value of $\mathcal{C}$. This will be further discussed in Section VI.

Algorithm 1 : A heuristic solution to the computation allocation problem.

```
1: Initialize $n_0 = k$ and $n_r = 0$ ($r = 1, ..., k-1$);
2: for Step $1, ..., \mathcal{C}$ do
3:     Go through $n_0, n_1, ..., n_{k-1}$ to find the smallest $r$ that satisfies $\forall r' \neq r$, $\frac{n_r}{n_{r+1}+1} \geq \frac{n_{r'}}{n_{r'+1}+1}$;
4:     Update $n_r = n_r - 1$ and $n_{r+1} = n_{r+1} + 1$;
5: end for
6: Output $([n_0, n_1, ..., n_{k-1}], [0, 1, ..., k-1])$;
```

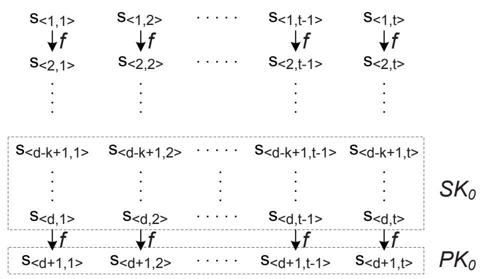


Fig. 5. The one-way chains with initial private and public keys.

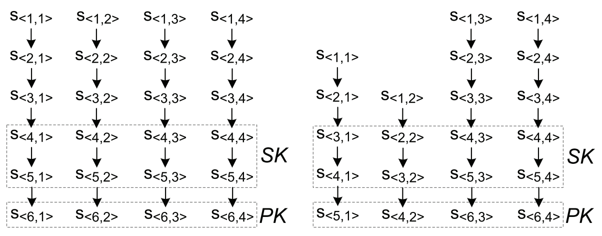
V. A MULTICAST AUTHENTICATION PROTOCOL

In this section, we design a multicast authentication protocol based on the proposed TSV scheme. One major challenge with OTS-based multicast authentication is the distribution of public keys. Similar with other work [20], [19], [28], we use one-way chains to distribute public keys efficiently. For convenience, we present our protocol based on LSHV, but it can be easily adapted to the general TSV.

A. The Protocol

Starting from t random values $s_{\langle 1,1 \rangle}, s_{\langle 1,2 \rangle}, ..., s_{\langle 1,t \rangle}$, the sender generates t one-way chains of length $d+1$ ($d \gg k$) and stores them as a series of keys (see Fig. 5). The initial public key is $PK_0 = (s_{\langle d+1,1 \rangle}, s_{\langle d+1,2 \rangle}, ..., s_{\langle d+1,t \rangle})$. We assume PK_0 can be distributed to each receiver securely, e.g., via unicast messages authenticated by HMAC. The initial private key SK_0 consists of the t k-element chain segments that are adjacent to PK_0, as shown in Fig. 5.

When a signature is generated (see Sec. III-C.2) and revealed, the key values included in the signature and those that can be generated by applying the one-way function over them are exposed. Thus, the sender refreshes its private key by replacing any exposed key values with their predecessors in the same chain. Also, a receiver updates its public key by replacing corresponding old key values with the new values from the signature. Figure 6 illustrates an example of key update.


s<1,1> s<1,2> s<1,3> s<1,4>
s<2,1> s<2,2> s<2,3> s<2,4>
s<3,1> s<3,2> s<3,3> s<3,4>
s<4,1> s<4,2> s<4,3> s<4,4>
s<5,1> s<5,2> s<5,3> s<5,4>
SK
s<6,1> s<6,2> s<6,3> s<6,4>
PK
s<1,3> s<1,4>
s<1,1> s<2,3> s<2,4>
s<2,1> s<1,2> s<3,3> s<3,4>
s<3,1> s<2,2> s<4,3> s<4,4>
s<4,1> s<3,2> s<5,3> s<5,4>
SK
s<5,1> s<4,2> s<6,3> s<6,4>
PK

(a) Initial state (b) After signing m (H(m)=1|2)

Fig. 6. An example of private key and public key update, where $t = 4$, $d = 5$, and $k = 2$. Since $H(m) = 1|2$, the signature of m includes key values $(s_{\langle 5,1\rangle}, s_{\langle 4,2\rangle})$. Note that the disclosure of $s_{\langle 4,2\rangle}$ means $s_{\langle 5,2\rangle}$ is also exposed, so both should be removed from the private key.

B. Public Key Distribution Cost

In LSHV, each message consumes $\frac{k(k+1)}{2}$ secret values. Thus, a total of $\frac{td}{k(k+1)/2}$ messages can be authenticated by one instance of chains, assuming the chains are depleted at the same time. Then, for a receiver, the additional communication cost caused by the transmission of the public key is on average $\frac{k(k+1)}{2d}l$ bits per message. When d is much larger than k^2, this additional cost is very small compared with the signature size, i.e., kl bits. For example, when $d = 1024$, $k = 8$ and $l = 80$, the additional cost is only 0.4B per message. This cost is even lower when other TSV configurations are used instead of LSHV.

C. Discussions

If the sender stores each element of the one-way chain, the storage cost is td elements. With $t = 1024$, $d = 1024$ and $l = 80$, the cost is 10MB. This cost might be too high when the sender is a field device. Recently, Coppersmith and Jacobsson [5] proposed an approach that can greatly reduce the storage overhead of one-way chains. For each chain, this approach stores $\log d$ precomputed elements. Then it uses one online one-way function computation to output an element required by the signature, and uses $\log d$ additional computations, which can be done offline after the message is sent, to relocate those precomputed elements. In this approach, the sender's storage cost can be reduced to $t \log d$ elements.

Some existing optimization techniques can also be applied to our scheme to further reduce the communication and computation overhead. For instance, salt chains [20] can be used to reduce the size of a key value (i.e., parameter l), and the technique of signing part of the hash value of a message instead of the whole value [28] can be used to reduce the number of key values in a signature (i.e., parameter k).

Some weaknesses with existing one-way chain based multicast authentication protocols [20], [19], [28] also apply to our protocol. For example, a loose time synchronization is required between the sender and the receivers. This is not a big issue in Smart Grid, which has already deployed time synchronization technologies.

VI. Evaluations and Practical Considerations

In this section, we evaluate TSV and discuss how to apply it for different multicast applications in Smart Grid.

A. Heuristic and Optimal Computation Allocation

We implemented the heuristic algorithm presented in Section IV on a Linux machine with 2.2GHz processor and 2GB memory. To investigate how close the signing cost of the heuristic solution is to the minimum, we also implemented an exhaustive search algorithm on the same machine to find the optimal solution to Formula 2. In our implementation an exhaustive search with very simple pruning can be completed in a few minutes for $k = 13$ and $\mathcal{C} = \frac{k(k-1)}{2} - 1 = 77$.

Table IV compares the signing cost obtained by our heuristic algorithm to the minimal signing cost at different values of $\mathcal{C}$. In most cases the heuristic results are identical to the minimum, which means our heuristic algorithm is very effective. Deviations from the minimum occur when $\mathcal{C}$ becomes large, but in these cases the minimal signing cost is low and the absolute difference is still small. Table IV also clearly shows the tradeoff between the signing and verification cost, which is adjusted by parameter $\mathcal{C}$. Besides, some consecutive values of $\mathcal{C}$ result in the same signing cost. In such cases, the smallest value of $\mathcal{C}$ should be selected.

B. Practical Optimizations with the Selection of $\mathcal{C}$

Table IV shows that the signing and verification cost can be adjusted by changing the value of $\mathcal{C}$. In reality, the multicast application can select the best $\mathcal{C}$ to achieve a specific optimization goal, e.g., to minimize the authentication delay or the total energy consumption.

1) Minimizing the Authentication Delay: Consider the delay caused by signature generation and verification. Let n_s and n_v denote the number of hash or one-way function calculations needed in signing and verification, respectively. Suppose the average time that the sender and receiver needs to complete one calculation is t_s and t_v, respectively. Then the total authentication delay is $t_d = n_s t_s + n_v t_v$. By going through the computation cost at different values of $\mathcal{C}$, we can find the best $\mathcal{C}$ with the minimum authentication delay.

Now we look at a multicast communication example where the sender is powerful in computation but the receivers have limited computing resources. Suppose $t_s = 0.2\ \mu s$, which is the time one SHA-1 computation takes on a powerful Linux machine with quad-core processor (2.0 GHZ, 4GB cache, 2GB memory) [28]. Also, suppose $t_v = 5$ ms, which is the time one SHA-1 computation takes on a resource-constrained platform (Crossbow TelosB with a 16-bit 8 MHz processor and 10KB RAM) [4]. When $t = 128$ and $k = 13$, from Table IV it is easy to find that the authentication delay is minimized at $\mathcal{C} = 11$ and the minimal delay is 138 ms.

2) Minimizing the Energy Consumption: Let ϵ_s and ϵ_v denote the average energy that the sender and receiver consume to calculate a one-way/hash function, respectively. Suppose the sender multicasts messages to N receivers. Then the total energy consumption of authentication is given by $\epsilon_s n_s + N\epsilon_v n_v$. Again, by going through different values of $\mathcal{C}$ we can find the best $\mathcal{C}$ with the minimum energy consumption.

C. Smart Grid Application Cases

We use demand-response and wide area protection as examples to show how TSV can be used in Smart Grid.

TABLE IV

THE SIGNING COST OF OUR HEURISTIC SOLUTION AND THE OPTIMAL SOLUTION AT DIFFERENT VALUES OF $\mathcal{C}$ WHEN $k = 13$. THE CONSTANT FACTOR $\frac{t^k}{t(t-1)\cdots(t-k+1)}$ OF THE SIGNING COST IS NOT COUNTED INTO THE LISTED VALUES.

$\mathcal{C}$	Min. Signing Cost	Signing Cost of Heuristic Solution	-	$\mathcal{C}$	Min. Signing Cost	Signing Cost of Heuristic Solution
0	6.2×10^9	6.2×10^9		20	1152	1152
1	4.8×10^8	4.8×10^8		21	864	864
2	8.0×10^7	8.0×10^7		$22, 23$	576	576
3	2.2×10^7	2.2×10^7		24	532	532
4	7.3×10^6	7.3×10^6		$25, 26$	288	432($\mathcal{C} = 25$), 288($\mathcal{C} = 26$)
5	2.2×10^6	2.2×10^6		27	192	288
6	9.7×10^5	9.7×10^5		$28 \sim 30$	144	288
7	4.8×10^5	4.8×10^5		31	96	288
8	2.4×10^5	2.4×10^5		$32, 33$	72	192($\mathcal{C} = 3225$), 96($\mathcal{C} = 33$)
9	1.2×10^5	1.2×10^5		$34 \sim 37$	48	96($\mathcal{C} = 34, 35$), 64($\mathcal{C} = 36, 37$)
10	60480	60480		38	32	32
11	34560	34560		$39 \sim 41$	32	32
12	25920	25920		$42 \sim 45$	32	32
13	17280	17280		$46, 47$	32	32
14	8640	11520		$48 \sim 54$	8	32($\mathcal{C} = 48, 49$), 16($\mathcal{C} = 50 \sim 54$)
15	5760	5760		55	6	16
16	4320	4320		$56 \sim 65$	4	16($\mathcal{C} = 56 \sim 59$), 8($\mathcal{C} = 60 \sim 65$)]
17	2880	2880		$66 \sim 77$	2	8($\mathcal{C} = 66, 67$), 4($\mathcal{C} = 68 \sim 73$), 2($\mathcal{C} = 74 \sim 77$)
18	2304	2304		78	1	1
19	1440	1440				

TABLE V

TSV VS. HORS IN DEMAND-RESPONSE AT THE SAME SECURITY LEVEL (80-BIT) AND BANDWIDTH COST ($k = 13$).

Scheme	Public Key Size	Signing (optimal)	Verification	Key Gen. (offline)
HORS	10KB	1	14	1024
TSV($\mathcal{C} = 6$)	1.28KB	1.8×10^6	20	384
TSV($\mathcal{C} = 7$)	1.28KB	9.2×10^5	21	384
TSV($\mathcal{C} = 8$)	1.28KB	4.6×10^5	22	384
TSV($\mathcal{C} = 9$)	1.28KB	2.3×10^5	23	512
TSV($\mathcal{C} = 10$)	1.28KB	1.2×10^5	24	512

1) Demand-Response: In demand-response, the receivers can be home appliances which are equipped with embedded communication devices. These embedded devices usually have very limited storage, and thus it is important to keep their storage overhead as low as possible.

Table V compares TSV (with some example values of $\mathcal{C}$) to HORS at the same security level (80-bit) and the same signature size ($k = 13$). As suggested in HORS [24], $l = 80$. In TSV, $t = 128$ and the storage cost of a receiver is 1.28KB. In HORS, $t = 1024$ and the storage cost is 10KB. Thus, TSV reduces the storage cost by a factor of 8 compared with HORS, and is more appropriate for demand-response.

Considering that in demand-response the sender is much more powerful but the receivers are constrained in computation, a relatively small $\mathcal{C}$ is recommended for TSV. Also, the best $\mathcal{C}$ can be selected for a specific optimization goal as discussed in Section VI-B.

2) Wide Area Protection: In this application, it is important to keep the signature size as small as possible since the data has to be multicasted frequently and a major part of the data is the signature and the real application data has only tens of bytes. Considering that the receiver is more powerful than the sender in computation, the special case LSHV (i.e., $\mathcal{C} = \frac{k(k-1)}{2}$) is recommended as the TSV configuration.

Table VI compares LSHV to HORS at the same security level (80-bit) and public key size $t = 1024$. In HORS $k = 13$ but in LSHV $k = 8$. Therefore, the signature size in LSHV is 80 bytes, which reduces that in HORS by 40%. Based on IEEE C37.118, the data frame includes 6 phasors, which means the frame size is 48 bytes. When LSHV is used for authentication, the total message size (i.e., data frame size plus signature size) is reduced by nearly 30% compared with that using HORS. Thus, our scheme is more appropriate due to the lower bandwidth cost.

TABLE VI

LSHV VS. HORS IN WIDE AREA PROTECTION AT THE SAME SECURITY LEVEL (80-BIT) AND STORAGE COST ($t = 1024$). SUPPOSE AN IEEE C37.118 DATA FRAME HAS 48 BYTES.

Scheme	Msg. Size	Signing	Verification	Key Gen. (offline)
HORS	178B	1	14	1024
LSHV	128B	1.028	37	8192

In LSHV signature verification has 23 more one-way function calculations. Since the receiver is powerful, the increased computations can be completed in a very short time, just a few microseconds according to a recent study [28]. The key generation also requires 7168 more one-way hash functions, but this can be done off-line.

VII. CONCLUSIONS

In this paper, we proposed an OTS scheme TSV to facilitate multicast authentication in Smart Grid. Compared with existing schemes, TSV generates much smaller signature and has much lower storage requirement. Thus, it is more appropriate for Smart Grid applications such as demand-response and wide area protection. The benefit is at the cost of increased computations in signature generation and/or verification. However, TSV can flexibly allocate the computations between the sender and receiver. We formulated the optimal computation allocation problem and proposed an effective and efficient heuristic algorithm to solve it. We evaluated TSV with implementations and case studies.

References

[1] IEC TC 57/WG 10-12. Iec61850 communication networks and systems in substations. 2003.

[2] Jurjen N. E. Bos and David Chaum. Provably unforgeable signatures. *CRYPTO*, 1992.

[3] Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. *ASIACCS*, 2006.

[4] Yu-Shian Chen, I-Lun Lin, Chin-Laung Lei, and Yen-Hua Liao. Broadcast authentication in sensor networks. using compressed bloom filters. *Proc. Int. Conf. on Distributed Computing in Sensor Systems*, 2008.

[5] D. Coppersmith and M. Jakobsson. Almost optimal hash sequence traversal. *Proc. of the 4th Conf. on Financail Cryptography*, 2002.

[6] U.S.-Canada Power System Outage Task Force. Final report on the august 14, 2003 blackout in the unitied states and canada: Causes and recommendations. 2004.

[7] W. Gao, Q. Li, B. Zhao, and G. Cao. Multicasting in delay tolerant networks: A social network perspective. *Proc. ACM MobiHoc*, 2009.

[8] R. Hasan, R. Bobba, and H. Khurana. Analyzing naspinet data flows. *PSCE*, 2006.

[9] IEEE. Ieee standard communication delivery time performance requirements for electric power substation automation. *IEEE Standard 1646-2004*, 2004.

[10] Ali Ipakchi and Farrokh Albuyeh. Grid of the future. *PAE-M*, 7, 2009.

[11] Chris Karlof, Naveen Sastry, Yaping Li, Adrian Perrig, and J. D. Tygar. Distillation codes and applications to dos resistant multicast authentication. *NDSS*, 2004.

[12] Leslie Lamport. Constructing digital signatures from a one way function. *Technical Report CSL-98, SRI International*, 1979.

[13] D. Liu and P. Ning. Multi-level utesla: Broadcast authentication for distributed sensor networks. *ACm Trans. in Embedded Computing Systems*, 3(4):800–836, 2004.

[14] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. *Proc. of MOBIQUITOUS*, pages 118–132, 2005. Washington, DC, USA.

[15] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos. Multicast authentication in fully adversarial networks. *IEEE Symposium on Security and Privacy*, 2004.

[16] Ralph C. Merkle. Secrecy, authentication, and public key systems. *UMI Research Press*, 1982.

[17] Ralph C. Merkle. A certified digital signature. *CRYPTO*, 1989.

[18] M. Mitzenmacher and A. Perrig. Bounds and improvements for biba signature schemes. *No. TR-02-02, Computer Science Group, Harvard University*, 2002.

[19] W. D. Neumann. Horse: an extension of an r-time signature scheme with fast signing and verification. *ITCC*, 2004.

[20] Adrian Perrig. The biba one-time signature and broadcast authentication protocol. *Proc. ACM CCS*, 2001.

[21] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. *IEEE Symposium on Security and Privacy*, 2000.

[22] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song. Efficient multicast packet authentication. *NDSS*, 2003.

[23] Michael O. Rabin. Digitalized signatures. *In Richard A. Demillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, Foundations of Secure Computation*, 1978.

[24] L. Reyzin and N. Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. *Proc. of the Australian Conference on Information Security and Privacy*, 2002.

[25] J. Slay and M. Miller. Lessons learned from the maroochy water breach. *Critical Infrastructure Protection*, 253, 2007.

[26] Dawn Song, D. Zuckerman, and J. D. Tygar. Expander graphs for digital stream authentication and robust overlay networks. *IEEE Symposium on Security and Privacy*, 2002.

[27] L. Su, B. Ding, Y. Yang, T. Abdelzaher, G. Cao, and J. Hou. ocast: Optimal multicast routing protocol for wireless sensor networks. *Proc. IEEE ICNP*, 2009.

[28] Qiyan Wang, Himanshu Khurana, Ying Huang, and Klara Nahrstedt. Time-valid one-time signature for time critical multicast data authentication. *Proc. IEEE INFOCOM*, 2009.

Appendix

Proof of Theorem 1: Since the proof is trivial for $k = 1$ and $k = 2$, in the following we only consider $k \geq 3$. For convenience, we call each integer $r \in [0, \mathcal{C}]$ a *rank*. Also, define *dominant rank* h as the rank with the largest number of signature elements, i.e., $\forall r \neq h$, $n_r \leq n_h$, and define *pivot rank* p as the rank with $n_p = 0$. To facilitate the proof, we first give two lemmas.

***Lemma** 1: When $\mathcal{C} < \frac{k(k-1)}{2}$, the dominant rank $h < k$.*

Proof: We prove by contradiction. Assume $h \geq k$. Suppose the ranks larger than h totally include j $(j \geq 0)$ signature elements.

Case 1: $j = 0$, which means h is the largest rank in the signature. Since $n_0 \leq n_h$, there are at least $k - 2n_h$ remaining ranks and each of them includes at least one but at most n_h elements. It is easy to see the number of invocations of f needed for these remaining ranks is:

$$n' \geq n_h \cdot 1 + n_h \cdot 2 + ... + n_h \cdot (k - 2n_h - 1) + 1 \cdot (k - 2n_h)$$

The total number of invocations of f for all h ranks is given by $n_{total} = n' + n_h h$. Let $\delta = n_{total} - \frac{k(k-1)}{2}$. Since $n_{total} \leq \mathcal{C} < \frac{k(k-1)}{2}$, we know $\delta < 0$. Substitute the expression of n':

$$\delta \geq \frac{1}{2}(n_h - 1)(k - \frac{4n_h + 3}{2})^2 + \frac{1}{8}(9 - n_h)$$

Since k is integer, we have $(k - \frac{4n_h+3}{2})^2 \geq \frac{1}{4}$. Thus, $\delta \geq \frac{1}{8}(n_h - 1) + \frac{1}{8}(9 - n_h) = 1$ which is a contradiction.

Case 2: $j > 0$. The partial signature that consists of the first $k - j$ elements falls into Case 1 if the total number of invocations applied to these elements (denoted by $\mathcal{C}'$) is smaller than $\frac{(k-j)(k-j-1)}{2}$. Let $\tilde{n}$ denote the total number of invocations of f applied to the j elements with rank larger than h. Since $\tilde{n} \geq j(k+1)$ and $\mathcal{C}' = \mathcal{C} - \tilde{n}$, we have:

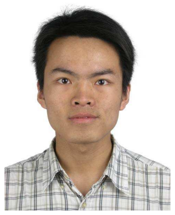
$$\mathcal{C}' < \frac{k(k-1)}{2} + \frac{j^2 + 3j}{2} - j(k+1) = \frac{(k-j)(k-j-1)}{2}$$

Then the proof reduces to Case 1. ■

***Lemma** 2: When $\mathcal{C} < \frac{k(k-1)}{2}$, if there is a rank $q \geq k$, there is also at least one pivot rank p and $\frac{\sqrt{1+8k}-1}{2} < p < k$.*

Proof: First we prove the existence of a pivot rank $p < k$ by contradiction. Assume there is no pivot rank $p < k$ with $n_p = 0$. Then the total number of invocations of f applied to the ranks smaller than k is at least $n' = 1 + 2 + ... + k - 1 = \frac{k(k-1)}{2}$, and the total number of invocations $n_{total} \geq n' + q > \frac{k(k-1)}{2} > \mathcal{C}$, which is a contradiction. Then we prove that $p > \frac{1+\sqrt{1+8k}}{2}$. Suppose the largest pivot rank is p. The number of invocations applied to ranks between $p+1$ and q is $n" = \frac{1}{2}(q^2 + q - p^2 - p) \geq \frac{k(k-1)}{2} + \frac{1}{2}(2k - p^2 - p)$. Since $n" \leq \mathcal{C}$, we get $2k - p^2 - p < 0$ and $p > \frac{\sqrt{1+8k}-1}{2}$. ■

To prove Theorem 1, we show that for any solution with some rank $r \geq k$ and $n_r > 0$, we can construct another solution which has no rank larger than or equal to k and has a smaller or the same signing cost. According to Lemma 2, there is at least one pivot rank and let p denote the largest pivot rank. We can remove one invocation of f out of each signature element whose rank is larger than p. Obviously, this will not increase the signing cost. After the removal, the ranks originally to the right of the pivot become equal to or larger than p. Then we add the removed invocations one by one to the left part of the pivot, i.e., the ranks $0, 1, ..., p-1$. In each step, add the invocation to the dominant rank h of the left part, and the signing cost of the left part does not increase as analyzed in Section IV-B. During the adding process, the total number of invocations included in the left part is smaller than $\frac{(p-1)(p-2)}{2}$. When $k \geq 3$, $p > \frac{\sqrt{1+8k}-1}{2} \geq 3$. Thus, Lemma 1 also applies to the left part by replacing k with $p-1$, which means $h < p-1$. After the invocation is added, no rank in the left part is larger than $p-1$. Thus, the signing cost of the right and left part change independently during the removing and adding process. Overall, the total signing cost does not increase. The above process can be repeated until the highest rank becomes smaller than k. The proof ends.



Qinghua Li received the B.E. degree from Xian Jiaotong University, China, and the M.S. degree from Tsinghua University, China, in 2004 and 2007, respectively. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at the Pennsylvania State University. His research interests include wireless networks and network security. He is a student member of the IEEE.

Guohong Cao received the BS degree from Xian Jiaotong University, China. He received the MS degree and PhD degree in computer science from the Ohio State University in 1997 and 1999 respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Professor. His research interests are wireless networks and mobile computing. He has published more than 150 papers in the areas of wireless sensor networks, wireless network security, vehicular ad hoc networks, data access and dissemination. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, and has served as program committee members of many conferences. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.