Secure Communication and Authentication Against Off-line Dictionary Attacks in Smart Grid Systems

Yongge Wang

UNC Charlotte, USA
`yongge.wang@uncc.edu`

Abstract. This paper studies the security requirements for remote authentication and communication in smart grid systems. Though smart card based authentication techniques have been a successful solution for addressing key management challenges in several cryptographic authentication systems, they may not be applicable to smart grid systems. For example, in order to unlock the credentials stored in tamper-resistant components (which could either be integrated in smart meters and collectors or be separate components that could be inserted into smart meters and collectors), one generally needs to input a password or PIN number to the smart meters or collectors. Since most smart meters and collectors are unattended, they could be maliciously modified or impersonated. Thus there is no trusted platform for the device owners or service provider agents to input the PIN number. Furthermore, the tamper resistant components (either integrated or separated) that hold the secret credentials could be easily accessed by an attacker and offline dictionary attacks could be easily mounted against these devices to retrieve the password or PIN number. In this paper, we review the security requirements for smart grid authentication systems and propose trust models for smart grid remote authentication systems. Finally, we propose secure authentication protocols within these trust models to defeat the common attacks such as offline dictionary attacks.

Keywords: password authentication, off-line attacks, smart grid authentication

1 Introduction

The smart grid is a secure and intelligent energy distribution system that delivers energy from suppliers to consumers based on two-way demand and response digital communication technologies to control appliances at consumers' homes to save energy and increase reliability. The smart grid improves existing energy distribution systems with digital information management and advanced metering systems. Increased interconnectivity and automation over the grid systems presents new challenges for deployment and management.

During 2011.02, more than 9,200 electric generating plants produced 312,334,000 megawatt-hours of electricity in the United States. Transmission lines distributed electricity to consumers in a 300,000 mile area. This power infrastructure was designed for performance and the integrated communications protocols were designed for bandwidth efficiency. However, cyber security was a low priority in the existing design of the power infrastructure systems. In order to transition from the current energy distribution infrastructure towards a smart grid, we have to overcome the challenges of integrating network-based security solutions with automation systems. Overcoming such challenges requires a combination of new and legacy components that may not have sufficient resources reserved to perform security functionalities (see, e.g., Wang [11, 14]).

One of the challenges for securing smart grid systems is to counter attacks on advanced meter infrastructures (AMI). In order for smart grid systems to securely manage remotely located smart meters and collectors, each meter or collector should contain a unique identifying credential. A straightforward suggestion could be to assign a unique PKI certificate for each meter and each collector. However, a careful analysis shows that this is infeasible in many scenarios. For instance, collectors are generally mounted in unattended areas and smart meters are generally mounted outside of consumers' houses (for various reasons). Thus both internal and external attackers have easy access to these devices and may try to recover the credentials stored in these devices. In order to address these challenges, secure credentials must be stored in tamper resistant components and these components could either be integrated into meters and collectors or be separated tokens that could be inserted into meters and collectors. Under this new application scenario, traditional cryptographic protocols may be easily broken (some examples will be presented in the next paragraphs and sections) and new protocols should be designed.

The tamper resistant components for smart meters and collectors could either be integrated into meter and collector design or could be separate tokens that are held by service provider agents. There are different security implications for different designs and different security models are needed correspondingly. It is also a common practice for an agent (or owner) to input passwords or PIN numbers to unlock the credentials stored in the tamper resistant components during a secure communication or authentication session. In the case that the tamper resistant component is a separate design (tokens), they look more like smart cards and secure smart card based protocols that have discussed in the literature may be applicable (see, e.g., Wang [12]) in the smart grid systems. In particular, in these models, the smart meters and collectors could be malicious and should not be considered as trusted platforms in the design. One of the major challenges in such kind of system design is that the PIN number or password for unlocking the credential in tamper resistant storage system may be recovered using offline dictionary attacks efficiently.

Numerous smart card based cryptographic protocols rely on passwords selected by users (people) for strong authentication. Since the users find it inconvenient to remember long passwords, they typically select short easily-rememberable passwords. In these cases, the sample space of passwords may be small enough to be enumerated by an adversary thereby making the protocols vulnerable to a *dictionary* attack. It is desirable then to design password-based protocols that resist off-line dictionary attacks (see, e.g., [17]).

The problem of password-based remote authentication protocols was first studied by Gong, Lomas, Needham, and Saltzer [3] who used public-key encryption to guard against off-line password-guessing attacks. In another very influential work (see, e.g., [17]), Bellovin and Merritt introduced Encrypted Key Exchange (EKE), which became the basis for many of the subsequent works in this area. These protocols include SPEKE and SRP (see, e.g., [17]). In models discussed in the above mentioned papers, we can assume that there is a trusted client computer for the user to input her passwords. In a smart grid authentication system, this assumption may no longer be true. The smart meters and collectors (which can be considered as smart card readers in certain scenarios) could be malicious and may intercept the user inputed passwords. Furthermore, a smart card could be stolen and the adversary may launch an off-line dictionary attack against the stolen smart card itself. Wang [12] has introduced several security trust models for smart card based remote authentication and designed secure protocols within these models. This paper will concentrate on the security trust models and protocols for smart grid systems.

In a practical deployment of smart grid based authentication systems, there may be other system requirements. For example, we may be required to use symmetric cipher based systems only or to use public key based systems. Furthermore, the system may also require that the server store some validation data for each user or the server do not store any validation (this can be considered as identity based systems). Furthermore, there may be other requirements such as user password expiration and changes.

There have been quite a number of papers dealing with smart card based remote authentications (see, e.g., [1, 2, 4, 6, 8, 9, 16]) and most of these papers present attacks on protocols in previous papers and propose new protocols without proper security justification (or even a security model). Recently, Wang [12] has carried out a systematic analysis on security models for smart card based remote authentication and designed several secure protocols within these models. In this paper, we will carry out similar systematic researches on security models for smart grid authentication systems.

2 Communication Channels in Smart Grid Systems

Figure 1 shows a typical deployment of Advanced Meter Infrastructure (AMI) systems, where the collector nodes accumulate data from advanced meters and then submit data to the headquarter computing systems. This is a typical data collection model in AMI systems and normally there is no direct physical communication channel between meters and headquarter computing servers. However, virtual secure communication channels could always be established between a meter and the headquarter servers when necessary as described in Figure 1. Furthermore, each of the meters and collectors may contain slots for a service provider agent to insert a secure token such as a smart card to initiate a sequence of secure activities such as remote authentication, meter/collector configuration, or secure communication between the service provider agent and headquarter computing systems via the meter/collector.

Though it is important to carry out research on general communication security and privacy preserving data collection in AMI systems, this paper concentrates on the following authenticated communication channels:

1. secure peer to peer authentication and communications among different nodes within the AMI systems. These nodes could be meters, collectors, or headquarters computing systems.
2. secure authentication of service provider agents or device owners via tokens (e.g., smart cards) inserted into meters and collectors

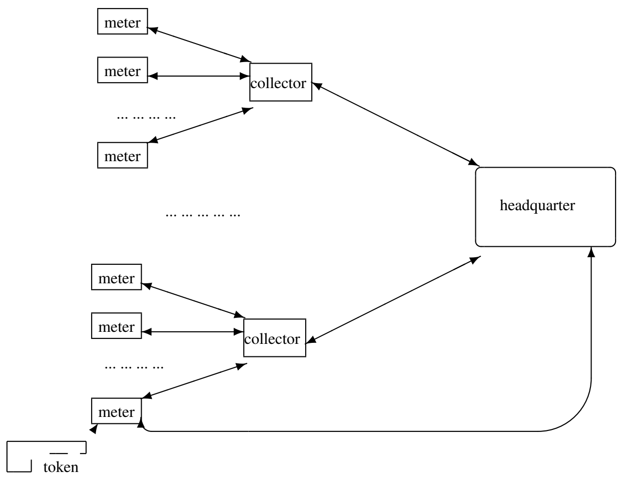

Fig. 1. A typical deployment of AMI systems

It is common for one to ask whether it is possible to use existing techniques such as Kerberos and PKI that have been successfully used in Internet environments to secure the communications among meters, collectors, and headquarter computing systems. The answer to this question is that we have to be very careful in using existing techniques. In a smart grid system, meters and collectors are normally installed at unattended areas. Thus it may be easy for an attacker to get long time access to a large amount of meters and collectors without being detected. In order to deploy Kerberos and PKI based cryptographic systems in smart grid systems, each node must hold a secure key (either a secret key for a symmetric cipher or a private key for a public key system). If secret keys in meters and collectors are not appropriately protected, an attacker could easily obtain them. Tamper resistant techniques are typically used to protect these keys. In order to shorten our notations in following discussions, we will only mention smart meters unless stated otherwise. The discussion applies to collectors or separate tokens that could be inserted into meters or collectors as well.

We use an example to show the challenges in the design of secure smart grid based authentication protocols using tamper resistant techniques. A traditional way to store or transfer the secret key for each user is to use a symmetric key cipher such as AES to encrypt user's long term secret key with user's password and store the encrypted secret key in meters/collectors (either in integrated tamper resistant components of the meters/collectors or in separate tokens to be inserted into the nodes). This will not meet our security goals against off-line dictionary attacks. For example, in an RSA based public key cryptographic system, the public key is a pair of integers (n, e) and the private key is an integer d. With the above mentioned traditional approach, the smart meter contains the value $AES_\alpha(d)$ in its tamper resistant memory space, where α is the user's password. If the adversary has access to the smart meter for certain time period, the adversary could feed a message (or challenge) m to the smart meter for a signature. The adversary needs to input a password in order for the smart meter to generate a signature. The adversary will just pick one α' from her dictionary and ask the meter to sign m. The meter will "decrypt" the private key as $d' = AES_{\alpha'}^{-1}(AES_{\alpha}(d))$ and return a signature $s' = m^{d'} \bmod n$ on m. Then the adversary only needs to check whether $s'^{e} \bmod n = m$. If the equation holds, the adversary knows that the guessed password α' is correct. That is, $\alpha' = \alpha$. Otherwise, the attacker will remove α' from the dictionary. Similar attacks work for Guillou-Quisquater (GQ), Fiat-Shamir, and Schnorr zero-knowledge identification schemes.

This example shows that the "off-line" dictionary attack in the smart grid or AMI environments is different from the traditional client-server based off-line dictionary attacks. One potential approach to defeat this kind of attacks is to set a counter in the smart meter. That is, the smart meter is allowed to sign at most certain number of messages, and then self-destroy it. However, this kind of protection may not be feasible since the smart meters are normally deployed for a long time of services (e.g., 30 years) and it is hard to appropriately choose optimal values for the counter.

3 Security models

First we provide a comprehensive list of attacks that a password-protected smart grid based authentication protocol needs to protect against. An ideal password-based authentication protocols for smart grid systems should be secure against these attacks and we will follow these criteria when we discuss the security of password-protected smart grid authentication protocols.

– **Eavesdropping**. The attacker may observe the communication channels.
– **Replay**. The attacker records messages (either from the communication channels between smart meters and service providers or between tamper resistant components and non-tamper resistant components within the smart meters) she has observed and re-sends them at a later time.
– **Man-in-the-middle**. The attacker intercepts the messages sent between the two parties (between user $\mathcal{U}$ and smart card C or between smart card C and servers $\mathcal{S}$) and replaces these with her own messages. For example, if she sits between the user and the smart meter, then she could play the role of smart meter in the messages which it displays to the user on the smart meter and at the same time plays the role of users to the smart meter. A special man-in-the-middle attack is the *small subgroup attack*. We illustrate this kind of attack by a small example. Let g be a generator of the group $\mathcal{G}$ of order $n = qt$ for some small $t > 1$. In a standard Diffie-Hellman key exchange protocol, the client C chooses a random x and sends g^x to the server $\mathcal{S}$, then $\mathcal{S}$ chooses a random y and sends g^y to C. The shared key between C and $\mathcal{S}$ is g^{xy}. Now assume that the attacker $\mathcal{A}$ intercepts C's message g^x, replaces it with g^{xq}, and sends it to $\mathcal{S}$. $\mathcal{A}$ also intercepts $\mathcal{S}$'s message g^y, replaces it with g^{yq}, and sends it to C. In the end, both C and $\mathcal{S}$ compute the shared key g^{qxy}. Since g^{qxy} lies in the subgroup of order t of the group generated by g^q, it takes on one of only t possible values. $\mathcal{A}$ can easily recover this g^{qxy} by an exhaustive search.

– **Impersonation**. The attacker impersonates the smart meter (using another smart meter that the attacker has access to or without using any smart meters) to authenticate to the remote service provider, impersonates a remote service provider to the smart meter, impersonates a token holder to the smart meter (if the smart grid is designed in such a way that a service provider agent uses some tokens to authenticate to smart meters) and a bogus smart meter impersonates an actual meter to a service provider agent with tokens.
– **Malicious smart meters**. The attacker controls the smart meter and intercepts a token holder inputed password. Furthermore, the attacker controls all of the communications between smart meter and the token holder, and all of the communications between smart meter and the remote server. For example, the attacker may launch a man in the middle attack between the token holder and the smart meter.
– **Stolen tokens**. The attacker steals a token from a service provider agent or an owner and impersonates the token holder to the remote server. In this case, the attacker could use the stolen token to impersonate the token holder with guessed passwords to the remote server with a limited time of failures since the server may disable the token from the server side after certain number of failures. If the attacker is allowed to use the token with guessed passwords to impersonate the token holder to the remote server for unlimited times of failures, then it will be considered as an on-line dictionary attack (a scenario that is not considered in this paper). However, the attacker is allowed for four kinds of further attacks that we will discuss in the following. One exception that we need to make in our security model is that we will not allow the attacker to control a malicious meter to intercept the token holder's password and then to steal the token from the token holder. There are four kinds of attackers based on the stolen token scenario:
  • *Tamper resistant token with counter protection*. The attacker cannot read the sensitive information stored in the tamper resistant memory within the stolen token. Furthermore, the attacker may only issue a fixed amount of queries to the token to learn useful information. The token will be self-destroyed if the query number exceeds certain threshold (e.g., the GSM SIM card V2 or later has this capability).
  • *Tamper resistant token without counter protection*. The attacker cannot read the sensitive information stored in the tamper resistant memory of th token. However, the attacker may issue a large amount of queries to the token to learn some useful information. For example, the attacker may setup a fake server and uses a malicious smart meter to guess the potential password.
  • *Token is not tamper resistant*. The attacker (with the token) may be able to break the tamper resistant protection of the token and read the sensitive information stored in the tamper resistant memory. In this case, the token looks more like a USB memory stick that stores the user credential with password protection. But still there is a difference here. In order for the user to use USB memory stick based credentials, the user needs the access to a trusted computer to carry out the authentication. However, one may assume that even if the token is not tamper resistant, it is not possible for a malicious smart meter to read the sensitive information on the token within a short time period (e.g., during the time that the token owner inserts the token into the meter for an authentication).

  - *Returned stolen token*. The attacker may steal the token from a token holder and carry out some analysis (e.g., mount some attacks based on the stolen token) and then return the token to the token holder without being detected by the token holder (that is, the token holder is not aware of the fact that the token has been lost for a while). The second author would like to thank Mr. Ding Wang for some discussions on related topics (note that Mr. Ding Wang is one of the authors for paper Wang et al [10]).
- **Password-guessing**. The attacker is assumed to have access to a relatively small dictionary of words that likely includes the secret password α. In an *off-line attack*, the attacker records past communications and searches for a word in the dictionary that is consistent with the recorded communications or carry out interaction with a stolen token without frequent server involvement (the attacker may carry out one or two sessions with server involved and all other activities without server involvement). In an *on-line attack*, the attacker repeatedly picks a password from the dictionary and attempts to impersonate $\mathcal{U}$, C, $\mathcal{U}$ and C, or $\mathcal{S}$. If the impersonation fails, the attacker removes this password from the dictionary and tries again, using a different password.
- **Partition attack**. The attacker records past communications, then goes over the dictionary and deletes those words that are not consistent with the recorded communications from the dictionary. After several tries, the attacker's dictionary could become very small.

We now informally sketch the definition of six types of security models.

1. **Type I**. The attacker $\mathcal{A}$ is allowed to watch regular runs of the protocol between a smart meter $\mathcal{R}$ (could be under the control of $\mathcal{A}$) and the server $\mathcal{S}$, can actively communicate with $\mathcal{R}$ and $\mathcal{S}$ in replay, impersonation, and man-in-the-middle attacks, and can also actively control a smart meter when a token holder inserts the token and inputs her password. Furthermore, the attacker may steal the token (e.g., smart card) from the token holder (if this happens, we assume that the attacker has not observed the user password from the previous runs of protocols) and issue a large amount of queries to the token using a malicious meter. However, we assume that the token is tamper resistant and the attacker could not read the sensitive data from the token. A protocol is said to be *secure* in the presence of such an attacker if (i) whenever the server $\mathcal{S}$ accepts an authentication session with $\mathcal{R}$, it is the case that the actual user $\mathcal{U}$ did indeed insert her token into $\mathcal{R}$ and input the correct password in the authentication session; and (ii) whenever a smart meter together with a token accepts an authentication session with $\mathcal{S}$, it is the case that $\mathcal{S}$ did indeed participate in the authentication session and the user $\mathcal{U}$ did indeed input the correct password.
2. **Type II**. The capability of the attacker is the same as in the Type I model except that when the attacker steals the token, it can only issue a fixed number of queries to the token using a malicious smart meter. If the number of queries exceeds the threshold, the token will be self-destroyed.
3. **Type III**. The capability of the attacker is the same as in the Type I model except that when the attacker steals the token, it will be able to read all of the sensitive data out from the token. But we will also assume that when a token holder inserts the token into a malicious smart meter for a session of authentication, the smart

meter should not be able to read the information stored in the tamper resistant section of the token. In other words, the token is not tamper resistant only when the attacker can hold the token for a relatively long period by herself. Another equivalent interpretation of this assumption is that the attacker may not be able to intercept the password via the smart meter and read the information stored in the token at the same time.
4. **Type I-r, II-r, II-r**. The capability of the attacker is the same as in the Type I or Type II or Type III models respectively except that we allow returned stolen tokens.

4 Secure authentication and key agreement protocols for Smart Grid Systems

4.1 Symmetric key based scheme: SSCA

In this symmetric key based smart grid authentication scheme SSCA, the server should choose a master secret β and protect it securely. Note that this master secret β could be different for different users (tokens). The Setup phase is as follows:

– For each user with identity C and password α, the token maker (it knows the server's master secret β) sets the token secret key as $K = \mathcal{H}(\beta, C)$ and stores $\mathcal{K} = \mathcal{E}_\alpha(K)$ in the tamper resistant memory of the token, where $\mathcal{E}$ is a symmetric encryption algorithm such as AES and $\mathcal{H}$ is a hash algorithm such as SHA-2.

In the SSCA scheme, we assume that the token has the capability to generate unpredictable random numbers. There are several ways for token to do so. One of the typical approaches is to use hash algorithms and EPROM. In this approach, a random number is stored in the EPROM of the smart card when it is made. Each time, when a new random number is needed, the token reads the current random number in the EPROM and hash this random number with a secret key. Then it outputs this keyed hash output as the new random number and replace the random number content in the EPROM with this new value. In order to keep protocol security, it is important for the token to erase all session information after each protocol run. This will ensure that, in case the token is lost and the information within the tamper resistant memory is recovered by the attacker, the attacker should not able to recover any of the random numbers used in the previous runs of the protocols. It should be noted that one may also use symmetric encryption algorithms to generate random numbers. Due to the reversible operation of symmetric ciphers, symmetric key based random number generation is not recommended for token implementation.

Each time when the user inserts her token into a meter (which could be malicious), the meter asks the user to input the password which will be forwarded to the token.

1. Using the provided password α, the token decrypts $K = \mathcal{D}_\alpha(\mathcal{K})$. If the password is correct, the value should equal to $\mathcal{H}(\beta, C)$. The token selects a random number R_c, computes $R_A = \mathcal{E}_K(C, R_c)$, and sends the pair (C, R_A) to the meter which will be forwarded to the server.

2. The server recovers the value of (C, R_c) using the key $K = \mathcal{H}(\beta, C)$ and verifies that the identity C of the token is correct. If the verification passes, the server selects a random number R_s, computes $R_B = \mathcal{E}_K(C, R_s)$, and sends (C, R_B, C_s) to the meter which forwards it to the token. Here $C_s = \text{HMAC}_{sk}(\mathcal{S}, C, R_s, R_c)$ is the keyed message authentication tag on $(\mathcal{S}, C, R_s, R_c)$ under the key $sk = \mathcal{H}(C, \mathcal{S}, R_c, R_s)$ and $\mathcal{S}$ is the server identity string.
3. The token recovers the value of (C, R_s) using the key $K = \mathcal{H}(\beta, C)$, computes $sk = \mathcal{H}(C, \mathcal{S}, R_c, R_s)$, and verifies the HMAC authentication tag C_s. If the verification passes, it computes its own confirmation message as $C_c = \text{HMAC}_{sk}(C, \mathcal{S}, R_c, R_s)$ and sends C_c to the server. The shared session key will be sk.
4. The server accepts the communication if the HMAC tag C_c passes the verification.

The protocol SSCA message flows are shown in the Figure 2

Fig. 2. Message flows in SSCA

$$\begin{array}{l} \underline{\text{Token}} \longrightarrow \underline{\text{Server}} : C, \mathcal{E}_K(C, R_c) \\ \underline{\text{Token}} \longleftarrow \underline{\text{Server}} : \mathcal{E}_K(C, R_s), C_s \\ \underline{\text{Token}} \longrightarrow \underline{\text{Server}} : C_c \end{array}$$

In the following, we use heuristics to show that SSCA is a secure authentication protocol in the Type I and Type II security models. If the underlying encryption scheme $\mathcal{E}$ and HMAC are secure, then eavesdropping, replay, man-in-the-middle, impersonation, password-guessing, and partition attacks will learn nothing about the password since no information of password is involved in these messages. Furthermore, a malicious meter can intercept the password, but without the token itself, the attacker will not be able to learn information about the secret key $K = \mathcal{D}_\alpha(\mathcal{K})$. Thus the attacker will not be able to impersonate the server or the token owner. When the attacker steals the token (but she has not controlled a meter to intercept the token owner password in the past), she may be able to insert the token into a malicious meter and let the token to run the protocols with a fake server polynomial many times. In these protocol runs, the attacker could input guessed password α'. The token will output $(C, \mathcal{E}_{K'}(C, R_c))$ where $K' = \mathcal{D}_{\alpha'}(\mathcal{K})$. Since the attacker has no access to the actual server (this is an off-line attack), the attacker can not verify whether the output $(C, \mathcal{E}_{K'}(C, R_c))$ is in correct format. Thus the attacker has no way to verify whether the guessed password α' is correct. In a summary, the protocol is secure in the Type I and Type II security models.

The protocol SSCA is not secure in the Type III security model. Assume that the attacker has observed a previous valid run of the protocol (but did not see the password) before steals the token. For each guessed password α', the attacker computes a potential key $K' = \mathcal{D}_{\alpha'}(\mathcal{K})$. If this key K' is not consistent with the observed confirmation messages in the previous run of the protocol, the attacker could remove α' from the password list. Otherwise, it guessed the correct password.

If we revise the attacker's capability in Type III model by restricting the attacker from observing any valid runs of the protocol before she steals the token, we get a new security model which we will call Type III′ model. We can show that the protocol SSCA is secure in the Type III′ model. The heuristics is that for an attacker with access to the value $\mathcal{K} = \mathcal{E}_\alpha(K)$, he will not be able to verify whether a guessed password is valid off-line. For example, for each guessed password α', she can compute $K' = \mathcal{D}_{\alpha'}(\mathcal{K})$. But she has no idea whether K' is the valid secret key without on-line interaction with the server. Thus the protocol is secure in the Type III′ security model.

Remarks: Modification of the protocol may be necessary for certain applications. For example, if the token identification string C itself needs to be protected (e.g., it is the credit card number), then one certainly does not want to transfer the identification string C along with the message in a clear channel.

4.2 Public key based scheme: PSCAb

In this section, we introduce a public key based token authentication scheme with bilinear groups: PSCAb, it is based on the identity based key agreement protocol from IEEE 1363.3 [5, 13].

In the following, we first briefly describe the bilinear maps and bilinear map groups.

1. G and G_1 are two (multiplicative) cyclic groups of prime order q.
2. g is a generator of G.
3. $\hat{e} : G \times G \to G_1$ is a bilinear map.

A bilinear map is a map $\hat{e} : G \times G \to G_1$ with the following properties:

1. bilinear: for all $g_1, g_2 \in G$, and $x, y \in Z$, we have $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$.
2. non-degenerate: $\hat{e}(g, g) \neq 1$.

We say that G is a bilinear group if the group action in G can be computed efficiently and there exists a group G_1 and an efficiently computable bilinear map $\hat{e} : G \times G \to G_1$ as above. For convenience, throughout the paper, we view both G and G_1 as multiplicative groups though the concrete implementation of G could be additive elliptic curve groups.

Let k be the security parameter given to the setup algorithm and $\mathcal{IG}$ be a bilinear group parameter generator. We present the scheme by describing the three algorithms: **Setup**, **Extract**, and **Exchange**.

Setup: For the input $k \in Z^+$, the algorithm proceeds as follows:

1. Run $\mathcal{IG}$ on k to generate a bilinear group $G_\subset = \{G, G_1, \hat{e}\}$ and the prime order q of the two groups G and G_1. Choose a random generator $g \in G$.
2. Pick a random master secret $\beta \in Z_q^*$.
3. Choose cryptographic hash functions $\mathcal{H}_1 : \{0,1\}^* \to G$, $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^*$, and $\pi : G \times G \to Z_q^*$. In the security analysis, we view $\mathcal{H}_1, \mathcal{H}_2$, and π as random oracles.

The system parameter is $\langle q, g, G, G_1, \hat{e}, \mathcal{H}_1, \mathcal{H}_2, \pi\rangle$ and the master secret key is β.

Extract: For a given identification string $C \in \{0,1\}^*$, the algorithm computes a generator $g_C = \mathcal{H}_1(C) \in G$, and sets the private key $d_C = g_C^\beta$ where β is the master secret key. The algorithm will further compute $g_\mathcal{S} = \mathcal{H}_1(\mathcal{S}) \in G$ where $\mathcal{S}$ is the server identity string, and store value $\left(C, g_\mathcal{S}, d'_C\right)$ in the tamper resistant token where $d'_C = \mathcal{E}_{\mathcal{H}_2(\alpha)}(d_C)$, α is token owner's password. and $\mathcal{E}$ is the encryption function that could be defined in one of the following ways:

1. $\mathcal{E}$ is a standard symmetric cipher such as AES
2. $\mathcal{E}_{\mathcal{H}_2(\alpha)}(d_C) = \text{AES}_{\mathcal{H}_2(\alpha)}(d_C) + i_0$ where $i_0 = \min\{i : \text{AES}_{\mathcal{H}_2(\alpha)}(d_C) + i \in G, i = 0, 1, \ldots\}$. For an inputed password α', d_C is computed as $\text{AES}^{-1}_{\mathcal{H}_2(\alpha')}(d'_C - i_0)$ where $i_0 = \min\{i : \text{AES}^{-1}_{\mathcal{H}_2(\alpha')}(d'_C - i) \in G, i = 0, 1, \ldots\}$.
3. $\mathcal{E}_{\mathcal{H}_2(\alpha)}(d_C) = d_C^{\mathcal{H}_2(\alpha)}$

Exchange: The algorithm proceeds as follows.

1. The token selects $x \in_R Z_q^*$, computes $R_A = g_C^x$, and sends it to the Server via the meter.
2. The Server selects $y \in_R Z_q^*$, computes $R_B = g_S^y$, and sends it to the token.
3. The token computes $s_A = \pi(R_A, R_B)$, $s_B = \pi(R_B, R_A)$, and $d_C = \mathcal{D}_{\mathcal{H}_2(\alpha')}(d'_C)$ where $\mathcal{D}$ is the decryption function and α' is the user inputed password. If d_C is not an element of G, the token chooses the value for sk as a random element of G_1. Otherwise, the token computes the value $sk = \hat{e}(g_C, g_S)^{(x+s_A)(y+s_B)\beta}$ as
$$\hat{e}\left(d_C^{(x+s_A)}, g_S^{s_B} \cdot R_B\right).$$
4. The token computes $K_1 = \mathcal{H}(sk, , R_A, R_B, C, \mathcal{S}, 1)$, $K_2 = \mathcal{H}(sk, , R_A, R_B, C, \mathcal{S}, 2)$, and sends value $C_C = \text{HMAC}_{K_1}(C, \mathcal{S}, R_A, R_B)$ to the server. K_2 is the shared secret.
5. Server computes $s_A = \pi(R_A, R_B)$, $s_B = \pi(R_B, R_A)$ and sk as
$$\hat{e}(g_C, g_S)^{(x+s_A)(y+s_B)\beta} = \hat{e}\left(g_C^{s_A} \cdot R_A, g_S^{(y+s_B)\beta}\right).$$
6. Server verifies whether C_C is correct. If the verification passes, server computes $K_1 = \mathcal{H}(sk, , R_A, R_B, C, \mathcal{S}, 1)$, $K_2 = \mathcal{H}(sk, , R_A, R_B, C, \mathcal{S}, 2)$ and sends the value $C_S = \text{HMAC}_{K_1}(\mathcal{S}, C, R_B, R_A)$ to the token. K_2 is the shared secret.
7. The token verifies the value of C_S.

The token should never export the value of sk to the meter during the protocol run. However, the token may need to export K_2 to the meter in certain applications.

The protocol PSCAb message flows are shown in the Figure 3

Fig. 3. Message flows in PSCAb

$$\begin{array}{l}
\underline{\text{Token}} \longrightarrow \underline{\text{Server}} : g_C^x \\
\underline{\text{Token}} \longleftarrow \underline{\text{Server}} : g_S^y \\
\underline{\text{Token}} \longrightarrow \underline{\text{Server}} : C_C \\
\underline{\text{Token}} \longleftarrow \underline{\text{Server}} : C_S
\end{array}$$

In the following, we use heuristics to show that PSCAb is secure in the Type I, Type II, and Type III security models. It should be noted that if the encryption function is chosen as a standard symmetric cipher such as AES, then PSCAb is only weakly secure in the Type III security model as follows. When the attacker has access to the value d'_C, she could remove those α' from her dictionary such that $\mathcal{D}_{\mathcal{H}_2(\alpha')}(d'_C)$ is not an element of G. In other words, PSCAb is secure in the type III security model only if the remaining dictionary is still large enough.

The security of the underlying identity based key agreement protocol WANG-KE [5, 13] is proved in [13]. Furthermore, the eavesdropping, replay, man-in-the-middle, impersonation, password-guessing, and partition attacks will learn nothing about the password since no information of password is involved in these messages. Furthermore, these attackers will learn nothing about the private keys d_C and β based on the proofs in [13]. For an attacker with access to the information d'_C (the attacker may read this information from the stolen token), she may impersonate the token owner to interact with the server. Since the attacker could not compute the correct value sk, she will not be able to generate the confirmation message C_C. Thus the server will not send the server confirmation message back to the attacker. In another word, the attacker will get no useful information for an off-line password guessing attack. Furthermore, even if the attacker has observed previous valid protocol runs, it will not help the attacker since the token does not contain any information of the session values x of the previous protocols runs.

Remarks: In the protocol PSCAb, it is important to have the token to send the confirmation message to the server first. Otherwise, PSCAb will not be secure in the Type III security model. Assume that the server sends the first confirmation message. After the attacker obtains the value d'_C from the token, she could impersonate the user by sending the vale R_A to the server. After receiving the server confirmation message, she will remove α' from her dictionary such that

$$sk' = \hat{e}\left(\mathcal{D}_{\mathcal{H}(\alpha')}(d'_C)^{(x+s_A)}, g_{\mathcal{S}}^{s_B} \cdot R_B\right)$$

is not consistent with the confirmation message $C_{\mathcal{S}}$.

4.3 Public key based scheme: PSCA

In the previous section, we presented a protocol PSCAb based on the identity based key agreement protocol WANG-KE. In this section, we briefly discuss a protocol based on the HMQV key agreement protocol [7]. Let g be the generator of the group $G_{\subset}$, q be the prime order of g, and h be a constant. In this case, the server and the token will both have public keys.

The server private/public key pair is (b, g^b). The token private/public key pair is (a, g^a). The data stored on the token is: $(a \times \mathcal{H}(\alpha), g^b)$. In the following, we use C and $\mathcal{S}$ to denote the client (token) and server identity strings respectively.

1. The token selects $x \in_R [1, q-1]$, computes $R_A = g^x$, and sends it to the server.
2. Server selects $y \in_R [1, q-1]$, computes $R_B = g^y$, and sends it to the token.
3. The token decrypts the private key a via the user inputed password, computes $\pi_A = \mathcal{H}(R_A, \mathcal{S})$, $\pi_B = \mathcal{H}(R_B, C)$, $s_A = (x + \pi_A a) \bmod q$, and the shared session key: $K_{\text{HMQV}} = (R_B \cdot (g^b)^{\pi_B})^{s_A h}$.
4. The server computes $\pi_A = \mathcal{H}(R_A, \mathcal{S})$, $\pi_B = \mathcal{H}(R_B, C)$, $s_B = (y + \pi_B b) \bmod q$, and the shared session key: $K_{\text{HMQV}} = (R_A \cdot (g^a)^{\pi_A})^{s_B h}$.

Remarks: Heuristics could be used to show that this protocol is secure in the Type I and Type II security models. However this protocol is not secure in the Type III security model. After the attacker obtains the value $(a \times \mathcal{H}(\alpha), g^b)$, the attacker could recover the password from $a \times \mathcal{H}(\alpha)$ and the token public key g^a. However, if g^a is only known to the server, then PSCA should be secure in the Type III model. We conjecture that it may be impossible to design HMQV based protocols that are secure in the Type III model if the public key of the token is available to the attackers.

4.4 Public key based scheme with password validation data at server: PSCAV

In previous sections, we discussed two protocols SSCA and PSCAb that the server does not store any password validation data. In this section, we discuss a protocol where the server needs to store password validation data for each token. One of the disadvantages of this kind of protocols is that if the token owner wants to change her password, the server has to be involved.

It should be noted that the password based remote authentication protocols that have been specified in the IEEE 1363.2 [5] are not secure in our models. The major reason is that the only secure credential that a client owns is the password. If the token owner inputs her password on an untrusted meter, the meter could just record the password and impersonates the client to the server without the token in future.

Before we present our scheme PSCAV, we briefly note that the protocol PSCAb in Section 4.2 can be easily modified to be a password protected token authentication scheme that the server stores user password validation data. In Section 4.2, the identity string for each user is computed as $g_C = \mathcal{H}(C) \in G$. For protocols with password validation data, we can use a different way to compute the identity strings. In particular, assume that the user $\mathcal{U}$ has a password α, then the identity string for the user will be computed as $g_C = \mathcal{H}(C, \alpha) \in G$ and the private key for the user will be $d_C = g_C^{\beta}$ where β is the master secret key. The value $(C, g_{\mathcal{S}}, \mathcal{E}_{\mathcal{H}_2(\alpha)}(d_C))$ will be stored in the tamper resistant token, and the value g_C will be stored in the server database for this user. The remaining protocol runs the same as in Section 4.2. We can call the above mentioned protocol as PSCAbV

Now we begin to describe our main protocol PSCAV for this section. Assume that the server has a master secret β (β could be user specific also). For each user with password α, let the user specific generator be $g_C = \mathcal{H}_1(C, \alpha, \beta)$, the value $g_C^{\mathcal{H}_2(\alpha)}$ is stored on the token, where $\mathcal{H}_2$ is another independent hash function. The value $g_C = \mathcal{H}_1(C, \alpha, \beta)$ will be stored in the server database for this user. The remaining of protocol runs as follows:

1. The token selects random x and sends $R_A = g_C^x$ to the server.
2. Server selects random y and sends $R_B = g_C^y$ to the token.
3. The token computes $u = \mathcal{H}(C, \mathcal{S}, R_A, R_B)$ where $\mathcal{S}$ is the server identity string, $sk = g_C^{y(x+u\alpha)}$, and sends $C_c = \mathcal{H}(sk, C, \mathcal{S}, R_A, R_B, 1)$ to the server
4. After verifying that C_c is correct, server computes $u = \mathcal{H}(C, \mathcal{S}, R_A, R_B)$, $sk = g_C^{y(x+u\alpha)}$, and sends the value $C_s = \mathcal{H}(sk, \mathcal{S}, C, R_B, R_A, 2)$ to the token.

The protocol PSCAV message flows are the same as for the PSCAb protocol message in the Figure 3 (but with different interpretation for the variables in the figure).

In the following, we use heuristics to show that PSCAV is secure in the Type I, Type II, and Type III security models. For the PSCAV protocol, the eavesdropping, replay, man-in-the-middle, token (client) impersonation, password-guessing, and partition attacks will learn nothing about the password due to the hardness of the Diffie-Hellman problem. For the attacker that carries out a server impersonation attack, it will receive the value R_A, and send a random R_B to the token. The attacker will then receive the token confirmation message C_C. The attacker may not launch an off-line dictionary attack on these information since for each guessed password α', it has no way to generate a session key sk' due to the hardness of the Diffie-Hellman problem. For an attacker with access to the information $g_C^{\mathcal{H}_2(\alpha)}$ (the attacker may read this information from the stolen token), she may impersonate the token owner to interact with the server. The attacker may send a random R_A to the server which could be based on $g_C^{\mathcal{H}_2(\alpha)}$, and receives a value R_B from the server. But it cannot compute the correct value for sk based on these information. Thus it could not send the confirmation message C_C to the server. Thus the server will not send the server confirmation message back to the attacker. In other words, the attacker will get no useful information for an off-line password guessing attack. Furthermore, even if the attacker has observed previous valid protocol runs, it will not help the attacker since the token does not contain any information of the session values x of the previous protocols runs.

Remarks: The attack described in the Remarks at the end of Section 4.2 could be used to show that it is important to have the token to send the confirmation message to the server first in the protocol PSCAV also.

5 Peer to peer communication in smart grid systems

In the previous sections, we presented smart grid authentication and communication protocols in the client-server model. In advanced smart grid systems, peer to peer communications among meters and collectors are essential also. In this section, we present a yellow page protocol for peer to peer authentication and communication in smart grid systems.

PKI and Kerberos systems are extensively used in Internet environments for peer to peer authentication and communication. However, Kerberos requires an online trusted server for 24 hours a day and PKI requires updated CRL (certificate revocation list) in real time. It is generally very expensive to maintain these services with guaranteed security. Since nodes of smart grid systems are relatively stable for a given period of time, we may design secure authentication and communication protocols based on yellow page services (e.g., LDAP servers). A yellow page service (e.g., LDAP server) is generally read-only and easy to maintain.

In the Yellow Page protocol YP, each node A has a secret key K_A which is stored in the tamper resistant component of node A (could be contained in a separate token such as smart card) and there is an online yellow page Y that stores the following entry for each ordered node pair $\langle A, B\rangle$ of the AMI system:

$$\langle A, B\rangle : \mathcal{E}_{K_A}(\mathcal{H}(K_B, A), B, A).$$

Note that K_A should be a random key with sufficient entropy and it could be protected with memorable password within the tamper resistant component of node A. If K_A does not have sufficient entropy, then off-line dictionary attacks are possible against the Yellow page protocol.

Each time when a node A wants to talk to a node B, the participating parties follow the following steps of the protocol:

1. A retrieves from the Yellow Page Y the entry $\langle A, B\rangle : \mathcal{E}_{K_A}(\mathcal{H}(K_B, A), B, A)$, and decrypts $\tau = \mathcal{H}(K_B, A)$.
2. A chooses a random value r and sends it to B.
3. B chooses a random value s and sends the following pair $(s, \mathcal{H}'(\mathcal{H}(K_B, A), r, s, 0))$ to A.
4. After receiving (s, σ) from B, A checks whether $\sigma = \mathcal{H}'(\tau, r, s, 0)$, and sends $\delta = \mathcal{H}'(\tau, s, 1)$ to Bob.
5. B checks whether $\delta = \mathcal{H}'(\mathcal{H}(K_B, A), s, 1)$.

The session key for nodes A and B to carry out subsequence communications is computed as $sk = \mathcal{H}'(\tau, s, A, B)$.

The full message flow for the yellow page protocol YP are shown in the Figure 4.

Fig. 4. Message flow in YP

$$\begin{array}{l}
\underline{\mathrm{A}} \longleftarrow \underline{\mathrm{Y}} : \mathcal{E}_{K_A}(\mathcal{H}(K_B, A), B, A) \\
\underline{\mathrm{A}} \longrightarrow \underline{\mathrm{B}} : r \\
\underline{\mathrm{A}} \longleftarrow \underline{\mathrm{B}} : (s, \mathcal{H}'(\mathcal{H}(K_B, A), r, s, 0)) \\
\underline{\mathrm{A}} \longrightarrow \underline{\mathrm{B}} : \mathcal{H}'(\tau, s, 1)
\end{array}$$

5.1 A note on a paper appeared in IEEE transaction on smart grid

It is always challenging to design secure authentication protocols appropriately. A student of the author of this paper published a paper [15] in the IEEE Transactions on Smart Grid without getting permission from Dr. Wang and included Dr. Wang as the co-author in that paper [15]. In this section, we briefly show that the protocol presented in [15] could be trivially broken.

In the "secure communication protocol" presented in [15], there is one trusted center T and several users (smart meters). Each user has a password. When Alice wants to talk to Bob, they will carry out the following protocol (note that both Alice and Bob could be smart meters or service provider stations):

– Alice sends "(Alice, Bob)" to T, Alice chooses a random r and sends "(r, Alice)" to Bob.
– T computes $\varepsilon = ENC(K_{alice}, \mathcal{H}(K_{bob}, Alice))$ and sends it to Alice, where ENC is a symmetric encryption scheme and K_{alice} and K_{bob} are Alice and Bob's passwords respectively.
– After Bob receives r from Alice, Bob chooses a random s and sends the value $(s, \sigma = \mathcal{H}'(\mathcal{H}(K_{bob}, Alice), r, s, 0))$ to Alice.

– Bob computes the session key

$$sk = \mathcal{H}'(\mathcal{H}(K_{Bob}, Alice), s, Alice, Bob)$$

– Alice decrypts

$$token = DEC(K_{alice}, \varepsilon) = \mathcal{H}(K_{bob}, Alice),$$

checks that $\sigma = \mathcal{H}'(token, r, s, 0)$, and sends the value $\delta = \mathcal{H}'(token, s, 1)$ to Bob

– Bob checks whether $\delta = \mathcal{H}'(\mathcal{H}(K_{bob}, Alice), s, 1)$

In the following, we present a trivial attack on the above protocol. Our attacks show that Carol can talk to Alice pretending to be Bob and Alice believes that she is talking to Bob though she is talking to Carol. In particular, the adversary Carol carries out the following steps of the attack:

– When Alice wants to talk to Bob, Alice sends the value "(Bob,Alice)" to T. At this stage, the adversary Carol intercepts this message and changes it to "(Carol, Alice)". T will reply $ENC(K_{Alice}, \mathcal{H}(K_{carol}, Alice))$ and Carol will forward this to Alice
– Alice sends "$(r, Alice)$" to Bob. Bob will not get this message though Carol (impersonating Bob) will get it.
– Carol (impersonating Bob) sends the value

$$(s, \mathcal{H}'(\mathcal{H}(K_{carol}, Alice), r, s, 0))$$

to Alice

– Alice sends $\mathcal{H}(token, s, 1)$ to Carol (impersonating Bob).

Now Alice is talking to Carol though Alice thinks that she is talking to Bob.

References

1. Y. Chen, J. Chou, and C. Huang. Comment on four two-party authentication protocols, 2010.
2. M. L. Das, A. Saxena, and V. P. Gulati. A dynamic id-based remote user authentication scheme. *IEEE Transactions on Consumer Electronics*, 50:629–631, 2004.
3. Li Gong, T. Mark, T. Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE J. Selected Areas in Communications*, 11:648–656, 1993.
4. T. Goriparthi, M. L. Das, and A. Saxena. An improved bilinear pairing based remote user authentication scheme. *Computer Standards and Interfaces*, 31:181–185, 2009.
5. IEEE 1363. Standard specifications for public-key cryptography, 2005.
6. W. S. Juang, S. T. Chen, and H. T. Liaw. Robust and efficient password-authenticated key agreement using smart cards. *IEEE Transactions on Industrial Electronics*, 55:2551–2556, 2008.
7. Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. Cryptology ePrint Archive, Report 2005/176, 2005. http://eprint.iacr.org/.
8. Y. Lee, J. Nam, and D. Won. Vulnerabilities in a remote agent authentication scheme using smart cards. In *LNCS: AMSTA, Vol. 4953*, pages 850–857. Springer-Verlag, 2008.
9. H. S. Rhee, J. O. Kwon, and D. H. Lee. A remote user authentication scheme without using smart cards. *Computer Standards and Interfaces*, 31:6–13, 2009.

10. Ding Wang and Chun-guang Ma. Robust smart card based password authentication scheme against smart card security breach. Technical report, Cryptology ePrint Archive, Report 2012/439, 2012. http://eprint. iacr. org/2012/439, 2012.
11. Yongge Wang. Cryptographic challenges in smart grid system security. In IEEE Smart Grid News Letters, December, 2012. available at http://smartgrid.ieee.org/december-2012/732-cryptographic-challenges-in-smart-grid-system-security.
12. Yongge Wang. Password protected smart card and memory stick authentication against off-line dictionary attacks. In *SEC*, pages 489–500, 2012.
13. Yongge Wang. Efficient identity-based and authenticated key agreement protocol. *Transactions on Computational Science*, 17:172–197, 2013.
14. Yongge Wang. Smart grid, automation, and scada systems security. In Yang Xiao, editor, *Security and Privacy in Smart Grids*, pages 245–268. CRC Press, July 2013.
15. Jinyue Xia and Yongge Wang. Secure key distribution for the smart grid. *IEEE Trans. Smart Grid*, 3(3):1437–1443, 2012.
16. T. Xiang, K. Wong, and X. Liao. Cryptanalysis of a password authentication scheme over insecure networks. *Computer and System Sciences*, 74:657–661, 2008.
17. Z. Zhao, Z. Dong, and Y. Wang. Security analysis of a password-based authentication protocol proposed to ieee 1363. *Theoretical Computer Science*, 352:280–287, 2006.