Building General-Purpose Security Services on EMV Payment Cards*

Chunhua Chen[1,**], Shaohua Tang[1,***], and Chris J. Mitchell[2]

[1] School of Computer Science and Engineering,
South China University of Technology,
Guangzhou 510640, China
chen.chunhua@mail.scut.edu.cn, csshtang@scut.edu.cn
[2] Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
c.mitchell@rhul.ac.uk

Abstract. The Generic Authentication Architecture (GAA) is a standardised extension to the mobile telephony security infrastructures that supports the provision of security services to network applications. We have proposed a generalised version of GAA which enables almost any pre-existing infrastructure to be used as the basis for the provision of generic security services, and have examined a GAA instantiation supported by Trusted Computing. In this paper we study another instantiation of GAA, this time building on the widely deployed EMV security infrastructure. This enables the existing EMV infrastructure to be used as the basis of a general-purpose authenticated key establishment service in a simple and uniform way, and also provides an opportunity for EMV-aware third parties to provide novel security services. We also discuss possible applications and issues of privacy and trust.

Keywords: GAA, EMV, key establishment, security service.

* This work was partially sponsored by the National Natural Science Foundation of China under Grant (No. U1135004 and 61170080), the Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011), the Guangzhou Metropolitan Science and Technology Planning Project (No. 2011J4300028), the Fundamental Research Funds for the Central Universities (No. 2009ZZ0035 and 2011ZG0015), the Guangdong Provincial Natural Science Foundation (No. 9351064101000003) and the High-level Talents Project of Guangdong Institutions of Higher Education (2012).

** The author is a PhD student at the South China University of Technology. This work was partially performed during a visit to the Information Security Group at Royal Holloway, University of London, sponsored by the Chinese Scholarship Council.

*** Corresponding author.

A.D. Keromytis and R. Di Pietro (Eds.): SecureComm 2012, LNICST 106, pp. 29–44, 2013.

1 Introduction

Almost any large scale network security system requires the establishment of some kind of a security infrastructure. For example, if network authentication or authenticated key establishment is required, then the communicating parties typically need access to a shared secret key or certificates for each other's public keys.

Setting up a new security infrastructure for a significant number of clients is by no means a trivial task. For example, establishing a public key infrastructure (PKI) for a large number of users involves setting up a certification authority (CA), getting every user to generate a key pair, registering every user and corresponding public key, and generating and distributing public key certificates. In addition, the ongoing management overhead is non-trivial, covering issues such as revocation and key update.

At the same time, there are a number of existing security infrastructures, in some cases with almost ubiquitous coverage. When deploying a new network security protocol it is therefore tempting to try to exploit one of these existing security infrastructures to avoid the need for the potentially costly roll-out of a new infrastructure.

This is by no means a new idea (see, for example, [17,18,19,21]). However, previous proposals have been ad hoc and application-specific. The alternative approach we consider here involves building a framework on top of an existing security infrastructure, which exploits the underlying infrastructure to enable the provision of general-purpose security services. For example, 3GPP has standardised the Generic Authentication Architecture (GAA) [4], which uses the mobile telephony security infrastructures (including those for GSM[1] and UMTS[2]) to provide a set of security services. A full description of 3GPP GAA is presented in [14]. Advantages of such a general approach include the usual benefits of a layered protocol architecture, including re-usability of applications across underlying infrastructures and simplified application development.

In previous work [6,7] we proposed a generalised version of GAA, which aims to enable almost any pre-existing infrastructure to be used as a basis for the provision of generic security services. A GAA instantiation supported by the Trusted Computing (TC) infrastructure has been described [6]. In this paper we build on the widely deployed EMV [9,10,11,12] (named after Europay, MasterCard, and Visa) security infrastructure, involving the chip-based EMV credit/debit cards deployed worldwide.[3] We define a GAA instantiation building on the EMV security services, which we call EMV-GAA.

The remainder of this paper is organised as follows. In section 2 we introduce a generalised version of GAA. In section 3 we provide an overview of EMV security services. In section 4 we give details of EMV-GAA. This is followed

[1] The Global System for Mobile Communications.

[2] The Universal Mobile Telecommunications System.

[3] Magnetic stripe cards are even more widely deployed for credit and debit purposes than EMV cards; however, they cannot store (or process) secret keys, and hence could not be used to support GAA.

by an informal security analysis in section 5. We analyse relevant privacy and security issues, and propose a modified scheme to address possible threats in section 6. We review related work in section 7, and discuss possible applications in section 8. In section 9 we draw conclusions.

2 Generic Authentication Architecture

We start by describing the generalised version of the GAA architecture on which we build our novel scheme. We introduce the main roles in the framework, its goals and rationale, and its two main procedures. This generalised GAA architecture was first described in [7], and was elaborated in [6].

2.1 Overview of GAA

As shown in Figure 1, the following entities play a role in GAA.

- The *Bootstrapping Server Function* (BSF) *server* B acts as a Trusted Third Party (TTP), and is assumed to have the means to access credentials belonging to a pre-existing security infrastructure. B uses the pre-established credentials to provide authenticated key establishment services to *GAA-enabled user platforms* and *GAA-aware application servers*. B uses its Fully Qualified Domain Name (FQDN) as its identifier Id_B.
- A *GAA-aware application server* S is assumed to have the means to establish a mutually authenticated, confidentiality- and integrity-protected channel with B, and an arrangement to access the security services provided by B. The means by which the secure channel between B and S is established is outside the scope of the GAA framework. In practice, this could be supported by well-established techniques such as SSL/TLS channels with both server and client side certificates, IPsec tunnelling, or some other appropriate 'virtual private network'. A permanent secure channel is also potentially beneficial from an efficiency viewpoint, because it can be reused for multiple protocol executions. The functionality of a *GAA-aware application server* is also referred to as the *Network Application Function* (NAF) *server*. S uses its FQDN as its identifier Id_S.
- A *GAA-enabled user platform* P is assumed to be equipped with credentials belonging to the pre-existing security infrastructure, and accesses the security services provided by B.

The user platform and the BSF server need to interact with the pre-existing security infrastructure, whereas the application server does not (it only needs to interact with the BSF server and the user platform). Also, the user platform and the application server do not need to have a pre-existing security relationship.

GAA provides a general purpose key establishment service for user platforms and application servers. As described below, GAA uses a two-level key hierarchy consisting of a master session key and server- and application-specific

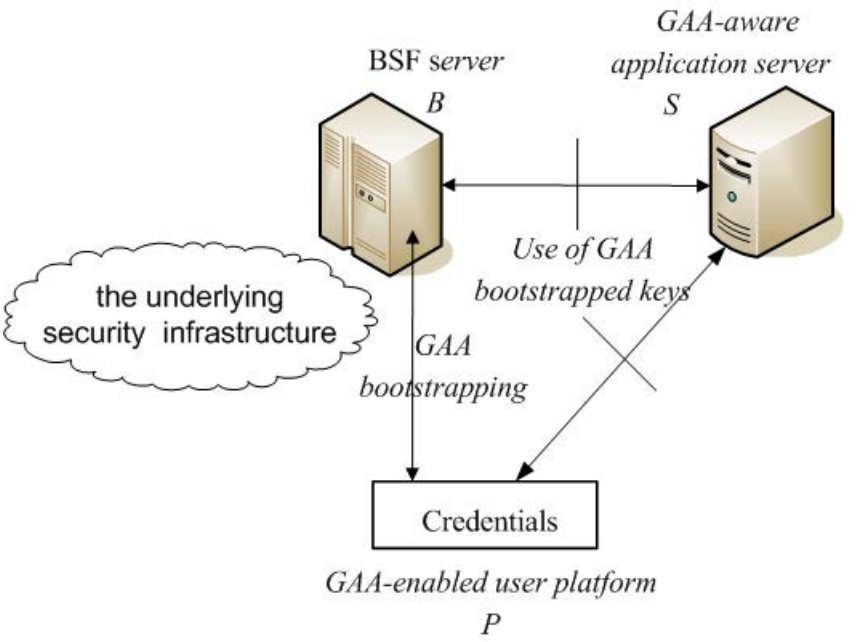

Fig. 1. GAA framework

session keys. The master session key is established using the pre-existing security infrastructure, and is not used directly to secure GAA-based applications. Instead it is used to generate the server/application-specific session keys using a *key diversification* function. By choosing a function with appropriate properties, it can be arranged that knowledge of a server/application specific session key will not reveal any information about the master session key or any other server/application-specific keys.

2.2 GAA Procedures

As we now describe, GAA incorporates two main procedures: *GAA bootstrapping* and *Use of bootstrapped keys*.

GAA bootstrapping uses the pre-existing security infrastructure to set up a shared master key MK between P and B. Also established is a Bootstrapping Transaction Identifier $B\text{-}TID$ for MK and the lifetime of this key. $B\text{-}TID$ must consist of a (statistically) unique value which identifies both an instance of *GAA bootstrapping* and Id_B.

The *Use of bootstrapped keys* procedure establishes a server/application-specific session key SK between P and S, using the master key MK shared by P and B. The procedure operates in the following way. P first derives a session key SK as:

$$SK = \mathrm{KDF}(MK, NAF\text{-}Id, \text{other values})$$

where KDF is a one-way *key diversification* function, and *NAF-Id* is an application-specific value consisting of Id_S and an identifier of the underlying application protocol. Other values may be included in the key derivation computation, depending on the nature of the underlying security infrastructure. P then starts the application protocol by sending a request containing *B-TID* to S. S submits the received *B-TID* and its own identifier *NAF-Id* to B to request the session key SK. Note that *B-TID* contains Id_B, so S knows where to send the request. As stated above, we require that S and B have the means to establish a mutually authenticated and confidential secure channel, and hence B can verify S against Id_S. If S is authorised, B derives SK from the MK identified by *B-TID*, and sends SK, its lifetime, and other relevant information to S via the secure channel. P and S now share SK, which they can use to secure application-specific messages.

Note that *key separation* is enforced by including *NAF-Id* as an input to the *key diversification* function. Other values used in the computation of SK could include identifiers for the GAA bootstrapping instance and the user platform.

3 EMV Security

In this section we provide a high-level overview of the main security features of the EMV payment system. We introduce the main roles in the system, the associated cryptographic keys and payment messages, and the processes relevant to this paper.

The EMV payment system involves five major interacting entities: a cardholder, an EMV payment card, a merchant terminal, an acquiring bank (the Acquirer) and a card issuing bank (the Issuer). The EMV specifications [9,10,11,12] define the interactions between an EMV smart card and a merchant terminal, as required to support financial transactions. Prior to engaging in such a transaction, the cardholder must complete an agreement with the Issuer, and be equipped with a chip-based EMV credit/debit card. The cardholder can then use this card to pay at merchant premises (EMV only supports transactions in which the cardholder is physically present, i.e. it does not support e-commerce or telephone transactions).

3.1 Transactions

An attempted EMV transaction can have a variety of outcomes; the transaction might be:

- approved offline;
- declined offline (by either the card or terminal); or
- sent for online approval by the card issuer.

We focus here on the case where the transaction is declined offline by the terminal, since we use the output of the card in this case as the basis of EMV-GAA bootstrapping, as described in section 4.2.

An EMV transaction that is declined offline involves the following steps. Many of the procedures involved in a typical transaction, including Data Authentication[4], Processing Restrictions, Cardholder Verification, Terminal Risk Management and Terminal Action Analysis, are omitted from this description, since they are not used in in EMV-GAA.

1. When a card is inserted into a terminal, the terminal first selects the EMV credit/debit payment application. Note that an EMV smart card could contain multiple applications, but will always contain an EMV payment application.
2. The terminal initiates Application Processing to start a new transaction session and to exchange information with the card. Note that in this mandatory step the Application Transaction Counter (ATC), a sequence number maintained by the card, is incremented.
3. The terminal reads Application Data from the card. During this mandatory step, the terminal acquires cardholder information (including the Primary Account Number (PAN) and PAN Sequence Number (PAN-SN)) from the card.
4. To decline the transaction, the terminal requests the card to generate an Application Authentication Cryptogram (AAC) (using the first GENERATE AC command [11, page 67]). The AAC is one example of an EMV-specific construct known as an Application Cryptogram (AC); an AC is a MAC computed on specific data using a key known only to a card and the card issuing bank.
5. The card performs Card Risk Management to protect the Issuer from fraud or excessive credit risk. Details of the card risk management algorithms within the card are specific to the Issuer, and are outside the scope of the EMV specifications.
6. The card performs Card Action Analysis to decide whether the transaction should be approved offline, transmitted online to be authorised by the Issuer, or declined offline. The card action analysis process is performed when the terminal issues the GENERATE AC command. Given that the card is requested to generate an AAC, the result of card action analysis is always to decline the transaction offline ([11, page 91]).
7. The card generates an AAC and returns it to the terminal. Details of this computation are described in section 3.2.
8. The terminal performs Completion, which ends processing of the current transaction.

3.2 AC Generation

In this section we provide further details of AC generation; we start by describing the secret keys involved.

[4] This procedure enables the terminal to verify the authenticity of the card. EMV specifies three modes of Data Authentication, namely Static Data Authentication (SDA), Dynamic Data Authentication (DDA) and Combined Data Authentication (CDA) [10, page 51]

The Issuer possesses an AC-specific 128-bit issuer master key, IMK_{AC}, used to generate the keys required to generate and verify ACs. When personalising a card, the Issuer uses the PAN and PAN-SN for this card as diversification information to derive a card-specific 128-bit Application Cryptogram master key MK_{AC} from IMK_{AC}. This key is installed in the card during personalisation.

When the card receives a GENERATE AC command, it first derives a 128-bit Application Cryptogram session key SK_{AC} from MK_{AC}, using the current ATC as diversification information. The card then uses SK_{AC} to produce the AC, a 64-bit cryptographic MAC computed as a function of a transaction-specific byte string formed by concatenating the following data items:

- values received from the terminal, including the Amount Authorized, the Transaction Date, and the Unpredictable Number (UN);
- values from within the card, including the ATC, which identify the current transaction.

The card returns the generated AC to the terminal, together with the Cryptogram Information Data (CID), the ATC, and other relevant data.

Depending on the result of the Card Action Analysis, the card will generate one of the following three types of AC:

- a Transaction Certificate (TC), if the transaction is approved offline;
- an Authorisation Request Cryptogram (ARQC), if an online authorisation is requested;
- an AAC, if the transaction is declined offline.

The CID contains two bits indicating the type of AC generated.

The Issuer needs to recompute the AC for verification purposes. This requires that the Account Identification Data of the card (i.e. the PAN and PAN-SN), the CID, the ATC, the UN provided by the terminal, and all other data objects used to compute the AC are transmitted to the Issuer. Using the received PAN and PAN-SN, the Issuer derives MK_{AC}, and from this obtains SK_{AC} using the received ATC. The Issuer then uses SK_{AC} to compute the particular type of AC indicated by the CID.

The EMV payment system makes use of a closed PKI to support Card Authentication. We use the term EMV security infrastructure to refer to the set of EMV cards possessed by cardholders, the Issuer servers, the associated secret keys, and the supporting PKI.

4 EMV-GAA

In this section we describe a possible means of using the EMV security infrastructure to support the generic version of GAA outlined in section 2.1, which we refer to as EMV-GAA. It is important to note that the scheme we propose here works with currently deployed EMV cards, using the existing card applications. That is, card re-issue (or card update) is not required.

4.1 Architecture

As shown in Figure 2, the following EMV-specific entities play a role in EMV-GAA.

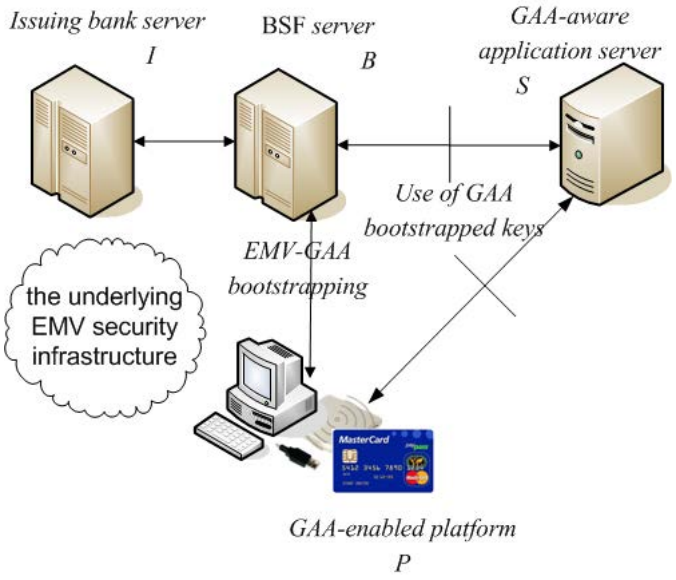

Fig. 2. EMV-GAA

- The Issuer I issues EMV-compliant cards, and possesses a master key IMK_{AC} which is used to derive the card-specific master keys MK_{AC}. I must be trusted for the purposes of supporting the EMV-GAA service by all parties using this service.
- The GAA-enabled user platform P incorporates a terminal T, an EMV-compliant card C (with an EMV debit/credit payment application), and the link between the two. To our knowledge, any EMV compliant card (SDA-, DDA-, or CDA-capable) could in principle be used to support EMV-GAA, since it does not make use of any of the Data Authentication procedures. T consists of a network access device and a card reader. A typical instantiation of T would be a Personal Computer (PC) with an attached or integrated card reader, where the card reader may or may not possess an integral key-pad (as shown in Figure 2). Alternatively, T could be a mobile device (such as a Personal Digital Assistant) capable of communicating with C, e.g. using Near Field Communication. We assume that T is equipped with a supporting application that implements the EMV-GAA bootstrapping protocol described below. This supporting application could be provided by a third party trusted for the purposes of delivering the EMV-GAA service by all parties using this service.

– The BSF server B connects to I via a secure communications channel to provide the GAA services. B must be trusted by all the parties used the EMV-GAA service.

We assume that the participating entities are connected via an open network such as Internet. Note that no assumptions are made about the security properties of these communications links.

4.2 Procedures

In this section we specify the EMV-GAA bootstrapping and the EMV-GAA Use of bootstrapped keys procedures.

The EMV-GAA bootstrapping protocol involves the following sequence of steps.

1. The terminal T sends an initial request to B for the bootstrapping of a master session key MK.
2. Upon receiving the request, B generates a random number R_B, associates it with a short time interval, and caches it[5]. B then sends R_B to T.
3. T commences communications with an EMV payment card C (if necessary first prompting the cardholder to insert C into the card reader), and selects the EMV credit/debit payment application in C. T next initiates a transaction session (which automatically causes the ATC to be incremented) and reads the card information, including the PAN, PAN-SN, etc. T will always declined the transaction, and will, accordingly, request C to generate an AAC by issuing the (first) GENERATE AC command. The UN sent with the GENERATE AC command is set to R_B, as selected in step 2. The other data items sent with the GENERATE AC command can be set to fixed values (in particular the 'Amount Authorised' can be set to 0).
4. C generates the AAC and returns the generated AAC, the CID data, the ATC, and the other values necessary to verify it. We refer below to the data sent with the AAC as M.
5. T ends the current transaction session with C.
6. T generates a random number R_T, associates it with a short time interval, and caches it. T then uses the AAC as a secret key K to compute a response RES as

$$RES = f(K, R_T, R_B, \mathrm{Id}_B, M).$$

The function f can be implemented in many ways. One possibility, which complies with clause 5.1.1 of ISO/IEC 9798-4 [15], is to instantiate f using HMAC [16] based on a suitable cryptographic hash function, where the

[5] B will clear R_B from its cache when the bootstrapping process completes or when R_B expires.

various inputs to f are simply concatenated prior to applying HMAC. That is, *RES* could be computed as:

$$RES = \mathrm{HMAC}_K(R_T||R_B||\mathrm{Id}_B||M)$$

where here and throughout || is used to denote concatenation. Note that the values $R_T||R_B$ and Id_B play the role of the nonce and the entity identifier, respectively, in the ISO/IEC 9798-4 protocol.

7. T sends PAN, PAN-SN, R_T, R_B, M and *RES* to B.
8. B checks that R_B is equal to the value selected in step 2. If not, B rejects the bootstrapping request and terminates the protocol.
9. B forwards PAN, PAN-SN and M to I.
10. I uses the information received from B to recompute the AAC, and then sends PAN, PAN-SN and the AAC back to B.
11. B uses the received AAC as a secret key K to recompute *RES*, and compares it with the *RES* received in step 7. If they do not match, B rejects the bootstrapping request; otherwise, B generates the master session key *MK* as

$$MK = \mathrm{KDF}(K, R_T, R_B).$$

B also sets the lifetime of *MK* (*LT*) in line with its operational policy, constructs the key identifier *B-TID* as a combination of R_B, R_T and Id_B, and stores PAN, PAN-SN, R_T, R_B, *B-TID*, *MK* and *LT*.

12. B computes

$$XRES = f(K, R_B, R_T, \mathrm{PAN}, \text{PAN-SN}, LT).$$

13. B sends R_B, R_T, *B-TID*, *LT* and *XRES* to T.
14. T checks that R_T is the same as the value it selected in step 6. T then recomputes *XRES*, and compares it to the value received from B. If either of these checks fail, the bootstrapping fails.
15. T computes *MK* in the same way as B, and then stores PAN, PAN-SN, R_B, R_T, *B-TID*, *MK* and *LT*.

During bootstrapping, the card-generated AAC is used as a secret key K shared by T and B to establish the master key *MK*. As defined in the EMV specifications (see [11], Table 33 in Annex A), an AAC contains only 64 bits. Hence, since it is derived from K, *MK* has at most 64-bit security. For a stronger *MK* (with 128-bit security), the protocol above requires the following changes.

- T must execute two separate declined offline EMV transactions with C; that is T carries out steps 3, 4 to 5 twice to obtain two AACs, and then concatenates them to form a 128-bit key K. In this case, R_B in step 2 needs to have length double that of the UN, and is used in two parts to initiate two GENERATE AC commands belonging to two EMV transactions.
- Accordingly, I must generate the two AACs for B in step 10.

Note that the ATC is incremented for every new EMV transaction. Since the ATC is used as diversification information in the computation of AAC, the two AACs above will almost certainly be different from each other.

In the EMV-GAA Use of bootstrapped keys procedure, P and S follow the procedure defined in section 2.2 to establish a server- and application-specific session key SK. The session key SK is derived (by B and P) as follows:

$$SK = \mathrm{KDF}(MK, R_T, R_B, \mathrm{PAN}, \mathrm{PAN\text{-}SN}, \mathit{NAF\text{-}Id}).$$

5 Informal Security Analysis

We now provide an informal security analysis of key aspects of the authentication and key establishment protocol used by the EMV-GAA bootstrapping procedure described in section 4.2. We consider a threat model in which an attacker $\mathcal{A}$ is able to observe and make arbitrary modifications to messages exchanged between B and P, including replaying and blocking messages as well as inserting completely spurious messages. This allows a trivial denial of service attack which cannot be prevented. Note that $\mathcal{A}$ is not allowed to compromise the implementations of B and P (including T and C); such attacks on system integrity cannot be prevent by the key establishment process, and are thus not addressed by the schemes we propose. We further assume that the communication channel between T and C is secure, since both devices are controlled by the user.

It is important to note that the security of EMV-GAA rests on the security of the underlying EMV security infrastructure; that is $\mathcal{A}$ is not allowed to compromise I or C.

The EMV-GAA bootstrapping protocol makes use of symmetric cryptographic techniques. The secret key K is an AAC, which can only be generated by the card C and the Issuer I (since it is a function of a key known only to them), and is securely transferred to T and B, respectively.

- *Entity authentication.* The protocol provides mutual authentication between B and T (strictly, C) using a cryptographic check function. B can verify the identity of T (strictly, C's PAN and PAN-SN); that is, the MAC generated by T on R_T, R_B and Id_B using K allows B to authenticate T (step 11). Similarly, T can authenticate B by verifying the MAC generated by B on R_B, R_T, PAN and PAN-SN (step 14). Messages exchanged in steps 2, 7 and 13 conform to the three-pass unilateral authentication protocol mechanism described in clause 5.2.2 of ISO/IEC 9798-4:1995 [15], in which the values R_B and R_T, generated by B and T respectively, serve as the nonces.
- *Confidentiality of the master session key MK.* MK is derived from K, which is shared by B and T. K is an AAC that can only be obtained by B and T. Hence, $\mathcal{A}$ cannot access MK under the assumed threat model.
- *Origin authentication.* This is achieved by B and T generating MACs on the exchanged messages using the key K. Integrity protection is also provided by the MACs. Hence, $\mathcal{A}$ cannot alter messages without being detected, since B and T will abort the bootstrapping procedure if any MAC verification fails (step 11 and 14).
- *Freshness.* R_B, generated by B, is included in the MAC sent to B in step 7; similarly R_T, generated by T, is included in the MAC sent to T in step 13.

Hence, $\mathcal{A}$ cannot replay messages to T or B, since B and T will abort the bootstrapping procedure if a received nonce is not fresh (step 8 and 14).
- *Key confirmation.* On receipt of the message in step 13, T can be sure that B has generated the MK during the current session. However, T does not confirm to B that it possesses MK. Note that $\mathcal{A}$ can block all the messages exchanged, and network errors might occur, and hence only T can be sure that it shares a fresh MK with B (until successful use of the key by T).
- *Key control.* The protocol is a key agreement process, that is B and T jointly control the inputs to the computation of MK (i.e. R_B and R_T).

6 Privacy and Security Issues

6.1 Threats

The EMV payment system is designed to be used in a closed (controlled) environment. A card terminal at a merchant typically provides a level of tamper-resistance, and is supplied by (or in conjunction with) the merchant's issuing bank. The terminal will be equipped with a pre-defined means of secure communication with the acquiring bank. By contrast, EMV-GAA operates in a more open environment. The terminal is user-controlled, and the communications with B and P are assumed to use the Internet or other public communications medium.

This change of environment gives rise to two main threats. Firstly, the scheme involves inserting the EMV card into a new type of terminal, which is itself a threat. A terminal can cause a card to perform a transaction, the precise nature of which is not apparent to the cardholder. Hence, a security threat arises whenever a card is inserted into any unauthorised terminal. Secondly, the PAN could be divulged to unauthorised entities and/or misused, including by a compromised terminal, by compromised software on a PC host, or by interception during transmission. The PAN can be regarded as Personally Identifiable Information (PII), and hence disclosure of the PAN is a privacy threat; it is also information which could be misused to conduct unauthorised transactions, and hence disclosure is also a security threat. We next consider the nature of these threats together with possible mitigations in greater detail.

Before using this service, it is likely that the cardholder will need to agree terms of use with the card issuer (and/or with the bootstrap server provider). This could include equipping the cardholder with a special card reader designed specifically for use with the EMV-GAA service—indeed, this is precisely what happens when cards are used with the CAP service, discussed below (and hence low-cost special-purpose card readers are clearly viable). This special reader could even be delivered as additional functionality in an enhanced CAP reader, further reducing deployment costs. The cardholder should in any case be advised never to insert his or her card into an unauthorised terminal. Part of the functionality of EMV-GAA could be built into the card reader (as is the case for CAP), thereby mitigating the threat of the card being forced to conduct unauthorised transactions. Finally we observe that the scheme we propose does

not require use of the Personal Identification Number (PIN) of the cardholder[6], further reducing the risk of an attacker being able to use a card to create illicit PIN-authorised transactions.

The use of a special purpose card reader also mitigates the risk of PAN disclosure at the cardholder site. PAN disclosure as a result of intercepted communications can be prevented by using the modified version of the scheme outlined below; the magnitude of the threat could also be reduced through the use of SSL on the connection between B and P, a standard precaution for security-related web connections.

6.2 A Modified Scheme

We now describe a minor modification to the bootstrapping procedure, designed to remove the need to transmit the PAN. The modified scheme requires the cardholder to register the EMV card C with the BSF server prior to use. As a result of the registration procedure, the BSF stores an association between a card-specific identifier Id_C and the pair (PAN, PAN-SN) for C. The identifier Id_C must be computed as a fixed function of data stored on the card, e.g. as $h(\mathrm{SSAD})$, where h is a cryptographic hash function and SSAD is the Signed Static Authentication Data, stored on the card and used in SDA.

The bootstrapping procedure is largely as described in section 4.2, except as follows.

- In step 3, T reads the SSAD from C, and computes Id_C for later use.
- In step 7, T sends Id_C to B instead of PAN and PAN-SN, and on receipt B uses Id_C to look up the values of PAN and PAN-SN.

7 Related Work

The Chip Authentication Program (CAP)[7] uses a EMV payment card in conjunction with a dedicated handheld device (the CAP reader) to produce one-time passwords (OTPs) for authenticating users and transactions in online banking. A dummy transaction with the card is started by requesting it to generate an ARQC, and after receipt of the ARQC the transaction is aborted. A decimal PIN is then computed as a function of the ARQC. Although the complete protocol details are not public, some information is in the public domain [8].

Urien [21] proposed the use of EMV payment cards to support the pre-shared key TLS protocol (TLS-PSK) [13]. The EMV TLS-PSK protocol provides mutual authentication, and could be used for on-line banking services. In EMV TLS-PSK, the pre-shared key identity is made of two parts: an identifier (EMV-ID) derived from parameters embedded in the card, and a set of cryptograms (i.e.

[6] The Cardholder Verification procedure in which the cardholder enters his or her PIN into the terminal is not used in EMV-GAA.

[7] CAP is a MasterCard brand; the corresponding Visa system is called Dynamic Passcode Authentication (DPA).

ARQCs). The pre-shared key is a fixed value, deduced from the EMV card content (EMV-PSK) and additional information. The EMV-PSK is set to $h(\text{SSAD})$, where h is a cryptographic hash function. The EMV-ID is set to $h(h(\text{SSAD}))$.

8 Applications

GAA offers a simple and uniform interface to generic security services which operate independently of the underlying security infrastructure. Application developers are thus able to use the services provided by this interface without having to understand the detailed operation of the underlying infrastructure, substantially simplifying the development task and reducing the risk of error. Moreover, this layered approach also enables the same application to operate over a variety of different underlying infrastructures in a transparent way.

In ongoing work we are examining ways in which a range of variants of the GAA service can be used to support an OTP system [5,7] for Internet applications. Such systems could, of course, be built using the EMV-GAA service. OTP systems supported by a range of GAA services could be deployed to enable the provision of ubiquitous OTP services for a large class of users. We are also developing ways of using GAA to build more general identity management solutions, including single sign-on schemes. Work along these lines has already been standardised for 3GPP GAA, notably supporting interworking with CardSpace, OpenID and Liberty [1,3]. We are also developing a way of enhancing the 'Pwd-Hash' mechanism [20] which builds on GAA service to give a user-centric single sign-on system.

The TLS-PSK protocol using the 3GPP GAA service (as supported by the mobile telephony authentication infrastructures) has been specified by 3GPP [2].

9 Conclusions

GAA is a framework that enables pre-existing security infrastructures to be used to provide general purpose security services, such as key establishment. We have shown how GAA services can be built on the EMV security infrastructure, complementing the previously proposed GAA schemes built on the mobile telephony authentication infrastructures and Trusted Computing. Use of EMV-GAA could constitute a potentially serious security and privacy threat (including the possibility of revealing the PAN to unintended parties). To mitigate the risk, we have proposed a modified scheme to avoid the need to routinely transmit the PAN across any network links.

EMV-GAA provides a way of exploiting the now very widespread EMV infrastructure for the provision of fundamentally important general-purpose security services. Of course, application-specific security protocols building on the infrastructure can be devised independently of any generic service and, indeed, there is a large and growing literature on such schemes. However, the definition of a standard GAA-based security service enables the EMV infrastructure to be exploited in a simple and uniform way, and it also provides an opportunity for EMV-aware third parties to provide novel security services.

References

1. 3G AMERICAS: Identity Management Overview of Standards & Technologies for Mobile and Fixed Internet (2009)
2. 3rd Generation Partnership Project (3GPP): Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS). Technical Specification TS 33.222, Version 9.1.0 (2009)
3. 3rd Generation Partnership Project (3GPP): Identity management and 3GPP security interworking; Identity management and Generic Authentication Architecture (GAA) interworking. Technical Report TS 33.924, Version 9.1.0 (2009)
4. 3rd Generation Partnership Project (3GPP): Technical Specification Group Services and Systems Aspects, Generic Authentication Architecture (GAA), Generic Bootstrapping Architecture. Technical Specification TS 33.220, Version 9.2.0 (2009)
5. Chen, C., Laitinen, P., Asokan, N., Mitchell, C.: Leveraging GAA for one-time password authentication from an untrusted computer (submitted)
6. Chen, C., Mitchell, C.J., Tang, S.: Building General Purpose Security Services on Trusted Computing. In: Chen, L., Yung, M., Zhu, L. (eds.) INTRUST 2011. LNCS, vol. 7222, pp. 16–31. Springer, Heidelberg (2012)
7. Chen, C., Mitchell, C., Tang, S.: Ubiquitous One-Time Password Service Using the Generic Authentication Architecture. Mobile Networks and Applications (to appear), `http://rd.springer.com/article/10.1007/s11036-011-0329-z`
8. Drimer, S., Murdoch, S.J., Anderson, R.: Optimised to Fail: Card Readers for Online Banking. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 184–200. Springer, Heidelberg (2009)
9. EMV: EMV Integrated Circuit Card Specifications for Payment Systems Version 4.2—Book 1: Application Independent ICC to Terminal Interface Requirements (June 2008)
10. EMV: EMV Integrated Circuit Card Specifications for Payment Systems Version 4.2—Book 2: Security and Key Management (June 2008)
11. EMV: EMV Integrated Circuit Card Specifications for Payment Systems Version 4.2—Book 3: Application Specification (June 2008)
12. EMV: EMV Integrated Circuit Card Specifications for Payment Systems Version 4.2—Book 4: Cardholder, Attendant, and Acquirer Interface Requirements (June 2008)
13. Eronen, P., Tschofenig, H.: Pre-shared key ciphersuites for transport layer security (TLS). Internet Engineering Task Force, RFC 4279 (Informational) (December 2005)
14. Holtmanns, S., Niemi, V., Ginzboorg, P., Laitinen, P., Asokan, N.: Cellular Authentication for Mobile and Internet Services. John Wiley and Sons (2008)
15. International Organization for Standardization, Genève, Switzerland: ISO/IEC 9798-4:1999, Information technology—Security techniques—Entity authentication—Part 4: Mechanisms using a cryptographic check function (1999)
16. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-hashing for message authentication. Internet Engineering Task Force, RFC 2104 (Informational) (February 1997)
17. Pashalidis, A., Mitchell, C.J.: Single Sign-On Using Trusted Platforms. In: Boyd, C., Mao, W. (eds.) ISC 2003. LNCS, vol. 2851, pp. 54–68. Springer, Heidelberg (2003)

18. Pashalidis, A., Mitchell, C.J.: Using GSM/UMTS for single-sign on. In: Proceedings of SympoTIC 2003, Joint IST Workshop on Mobile Future and Symposium on Trends in Communications, pp. 146–152. IEEE Press (2003)
19. Pashalidis, A., Mitchell, C.J.: Using EMV Cards for Single Sign-On. In: Katsikas, S.K., Gritzalis, S., López, J. (eds.) EuroPKI 2004. LNCS, vol. 3093, pp. 205–217. Springer, Heidelberg (2004)
20. Ross, B., Jackson, C., Miyake, N., Boneh, D., Mitchell, J.C.: Stronger password authentication using browser extensions. In: Proceedings of the 14th USENIX Security Symposium, pp. 17–32. USENIX Association (2005)
21. Urienand, P.: Introducing TLS-PSK authentication for EMV devices. In: Proceedings of CTS 2010, International Symposium on Collaborative Technologies and Systems, pp. 371–377. IEEE Press (2010)