Privacy Enhanced Device Access

Geir M. Køien

University of Agder
Department of ICT, Grimstad, Norway
geir.koien@uia.no

Abstract. In this paper we present the case for a device authentication protocol that authenticates a device/service class rather than an individual device. The devices in question are providing services available to the public. The proposed protocol is an online protocol and it uses a pseudo-random temporary identity scheme to provide user privacy.

Keywords: Authentication, Key derivation, Device access, Internet-of-Things, Security protocol, Privacy, Public access.

1 Introduction

1.1 Background

In this paper we present the case for a lightweight device access protocol that authenticates a device/service class rather than an individual device. The background for this is the observation that, for a human user (through his/her proxy device), access is often towards a service rather than any specific device.

A typical public Internet-of-Things (IoT) device is often a nameless device as seen from the human users perspective. An IoT device typically consist of an embedded computing platform with various sensors and actuators, and it has IP-connectivity (wired/wireless). There is little incentive for the user to know the identity (serial number or similar) of the device per se. Except of course, that this is often a prerequisite for connectivity. We assume that the IoT devices are accessible via a “broadcast” type of channel. Thus, the user cannot ascertain a device merely by means of physical connectivity (i.e. by fixed line cabling).

The devices are managed by an administrative entity and it will issue device access credentials to the user. Depending on the type of service requested it should be possible to access multiple devices during the same service period.

1.2 Use-Cases

We propose three example services where our device/sevice-access protocol could be deployed. Example cases:

- **C1: Parking Permit**
  A parking lot entrance is controlled by an IoT device. The permit may apply to any parking facility operated by the granting authority or for a a single location. The permit may be for a single access or for a pre-defined period.

- **C2: Auto Wash Machine**
  The auto wash machine is controlled by an IoT device. Washing rights can used for any auto wash machine operated by the same management. The washing right may be for one or more washes and for different types of washing.
- **C3: Wi-Fi Login Services**
  The user may buy access right for a period (`D` days, `M` months, ...) with the Wi-Fi operator. The access right could apply to all Wi-Fi access points within the operators network.

1.3 Outlining the Problem

We here list some aspects that must be considered for our device access protocol.

- *Flexible service authorization during access*
  Flexibility is needed when defining what an "access right" amounts to. Time period or a defined no. of access events? Specific device or any device? It may be beneficial to let the device operator handle the service authorization part.
- *Online vs. offline*
  Is the device operator to be online or offline during device access? There are pros and cons to both offline and online models. Requirements for authentication, authorization and accounting point towards online models, while communications cost and possibly user privacy will benefit from an offline model.
- *No Group Keys*
  As stated we want access to be for a service type rather than for any specific device. However, we do not want a scheme with a symmetric group key for a all devices with the given service type, since we consider group keys to be a security and privacy liability. We argue here that the IoT devices in question will be widely distributed and that while physical protection will be in place, it will nevertheless be limited. Then it is important that the compromise of one device does not unduely lead to compromise of other devices.
- *User privacy*
  We insist that the user be given credible privacy protection with respect to identity privacy and location privacy. This requirement is important not only toward external parties, but is also of importance internally. In particular, we want to limit the IoT devices ability to compromise the privacy of the users.
- *A Preference for Simplicity*
  The device/service access protocol should be as simple and lightweight as possible with respect to computational and communication needs.

2 Reference Architecture and Assumptions

2.1 Principal Entities

We will have three different types of principal entities:

– `OPR` : An administrative entity which owns and/or operates the devices.
– `USR` : The user is represented by a proxy device (possibly a a smart phone).
– `DEV` : The generic IoT device, which will provide/facilitate services.

As is customary we assume that we have honest principals, that will faithfully conduct their business as intended. However, we will also have a defined intruder, which we denote `DYI`, in our system and this intruder is free to masquerade as a principal. More on our intruder in subsection 3.1.

2.2 Generic Architecture

Figure 1 depicts the architecture, in which the user (`USR`) has gained access to a device (`DEV 3`). We chose to model a one-to-one correspondence between a service and the associated provider. Obviously, multiple provider may offer the same basic service and one provider may offer multiple distinct services. If one models the different consumers, the different services and the different providers as distinguishable, over the interfaces, then a richer model can be defined, but the essential properties will not differ in the logical layout of the models. For sake of simplicity we only depict a simplified model in figure 1.

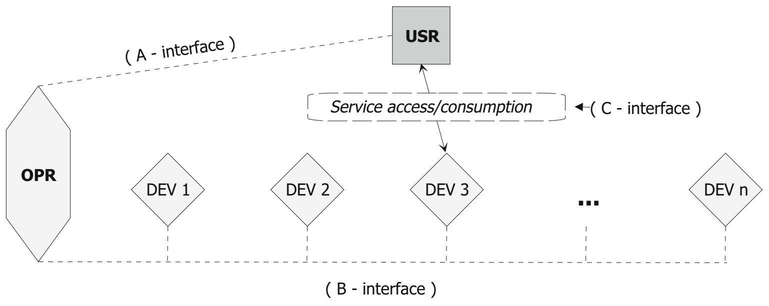

Fig. 1. Simplified Architecture

2.3 Interfaces and Channels

Between the principal entities we define the following interfaces:

– `A`-interface: Between `OPR` and `USR`
– `B`-Interface: Between `OPR` and `DEV`
– `C`-interface: Between `USR` and `DEV`

Each interface is associated with a logical channel. For sake of simplicity, the channel is named after the interface.

The A-interface/A-channel is used for service agreement. This includes all agreements between the user and the operator, and will include exchange of security credentials to facilitate service access between the user and the devices. We shall not assume that the A-interface is operational during service access.

The B-interface/B-channel is for communication between the device and the operator. This includes device initialization and it may include session authorization during user device access.

The C-interface/C-channel is defined between the user and the devices. The associated C-channel is established during service access setup and will be available during service consumption.

2.4 Communication Security Assumptions

The A-channel is assumed to be an authenticated and fully secured channel. Security for the A-channel is assumed to be pre-arranged and the actual setup and agreement of security credentials and key material for the A-channel is considered outside the scope of this paper.

The B-channel is assumed to be an authenticated and fully secured channel. The operators own/manages the devices so it seems reasonable to assume that security credentials and key material is simply distributed by the operator to the devices during device deployment.

The C-channel will need to be mutually authenticated, with respect to service consumption rights, and it will need to be protected. The protection in question should certainly include data integrity protection. Digital Rights Management and user privacy may dictate data confidentiality too, and we claim that it in general is best to assume data confidentiality as a requirement.

2.5 Computational Performance

We do not expect the computational requirements, with respect to provisioning access, to be prohibitive if one mainly designs the security and privacy part of the protocol around symmetric-key cryptographic primitives.

Neither do we in general anticipate that careful and discriminate use of normal public-key primitives are problematic. However, that assumption may be at odds with some devices, like passive RFID devices or devices with limited battery power etc, and for those cases one may find the requirements to be computationally exhausting for frequent access. Thus there is a case for making the access protocol as lightweight as possible with respect to computational requirements.

2.6 Assumed Trust Relationships

We have the following trust relationships:

- **Trust** $\mathtt{OPR} \rightleftarrows \mathtt{DEV}$
  The operator has full security jurisdiction over the devices. Devices are plentiful and may become corrupted or may otherwise fail. The operator therefore

cannot afford to trust the device too much. Trust has to be contained such that compromise of one device would not unduely lead to compromise of other devices. The devices must trust the operator fully.

- **Trust** `OPR`$\rightarrow$`USR`
  This relationship is governed and limited by contractual agreement.
- **Trust** `USR`$\rightarrow$`OPR`
  The user must trust the operator with respect to service purchase and consumption. Privacy: The user does not fully trust the operator with information regarding service consumption (time/location etc), but must trust the operator with payment information and possibly with identity information.
- **Trust** `USR` $\rightleftarrows$ `DEV`
  There is no a priori security trust between these entities. All trust must be through operator mediation. The established trust depends on trust transitivity. Transitive trust is indirect and we assume it to be weaker than direct trust. The trust level between the user and device is affected by whether the operator is online or not. For the offline case, one cannot know whether the operator still authorizes the access. The privacy trust that a user may have in a device is strictly limited to the needs for service access.

3 Intruder Model and Intruder Mitigation

3.1 Intruder Model

The classical intruder model is the well known Dolev-Yao (DY) intruder model [1]. In this model the intruder can intercept, modify, delete and inject messages at will. The intruder will have complete history information of all previous message exchanges and it can (and will) use this to the full extent to attack the system. It may also masquerade as a legitimate principal. The DY intruder is also a privacy intruder and it will exploit leakage of privacy sensitive data to the full extent.

The DY intruder is a very powerful adversary, but it has its limitations. For instance, it cannot physically compromise a principal and it cannot actually break cryptographic primitives (but it can exploit improper use). It may appear somewhat contrary that it cannot break "weak" cryptographic primitives, but this is nevertheless the standard assumption.

In the general case for an IoT environment one must expect devices to break down or otherwise be unable to maintain their physical integrity. Thus, with statistical validity, one must expect devices to be compromised and secrets to be exposed. The intruder will, by definition, exploit these shortcoming and will thus learn whatever secrets the IoT device contains, including privacy sensitive user data. The exploitation will potentially continue for the lifetime of the device.

However, while we advocate using the DYI model for analysis we still want to be realistic with respect to real-life intruders. These intruder will rarely be capable of using all available knowledge, and so pose less of a danger to the system. On the other hand, real-world intruder are capable of breaking "weak" crypto primitives and crypto system with short keys etc.

3.2 Compromise Containment

The compromise of one or more devices should not give the intruder an advantage in compromising other principal entities, whether devices, operators or users. Therefore, we require that secrets and sensitive data (key material, identities etc) stored at the devices should not unduely permit the DYI to compromise other principals.

With respect to cryptography we require the key sizes etc to be "sufficiently long", and we here advise adhering to the ECRYPT II recommendations [2].

4 Security Requirements and Privacy Aspects

4.1 Security Requirements

The user must verify that the device is authorized to provide services (by the operator). Correspondingly, the device must be assured that the user has obtained the rights to access the indicated service. The requirements are fairly easy to satisfy if the operator is online during the device access. If the B-interface is unavailable during device access, then the access protocol must provide assurance that the access once was sanctioned by the operator.

A suitable set of session keys must be agreed for the device access and service access procedures.

4.2 The Need for Privacy

The user identity, whether associated or emergent, should not be disclosed to any unauthorized parties. Also, the user location should not unduely be disclosed and tracking of the user should not be permitted. Data privacy, in conjunction with service payload and with identifying the consumed service must also be provided.

We note that the devices are not fully trusted with respect to user privacy. That is, the devices (`DEV`) are partially an unauthorized party. The operator (`OPR`) must be trusted by the user `USR`, but we still want the operator knowledge of privacy sensitive data to be limited to a minimum.

5 Online Model vs. Offline Model

5.1 Online Model

For an online case one can simplify the the protocol structure significantly and provide an assurance level that simply is not attainable with offline protocols [7]. The operator provides assurance and can forward key material to the device upon request, and thus the user and the device need not have complete credentials to start off with. This makes it is feasible to carry out all operations with symmetric-key primitives. This is an advantage when designing a computationally lightweight protocol solution.

5.2 Offline Model

To achieve satisfactory security for the offline case is not easy with the requirements that the device should not need to know the identity of the user.

To achieve the given goals, and to avoid using group keys, it is necessary to use asymmetric crypto primitives. To keep things simple we suggest that the operator issue short-lived digital certificates to the user. These digital certificates could be service-specific. The device verifies that the user had a legitimate "access certificate" and grants access accordingly. Since there already exists several suitable public-key/digital certificate protocols we suggest that one then deploys such protocol instead of inventing a new protocol. We suggest using TLS (TLS v1.2, RFC 5246 [3]). Certificate revocation must be checked (RFC 3280 [4]), but in an offline model this information may be obsolete.

6 Outline of Online Solution

The following section outlines the "Privacy Enhanced Lightweight Device Authentication" (PELDA) protocol.

6.1 Protocol Outline

Figure 2 depicts the main PELDA protocol.

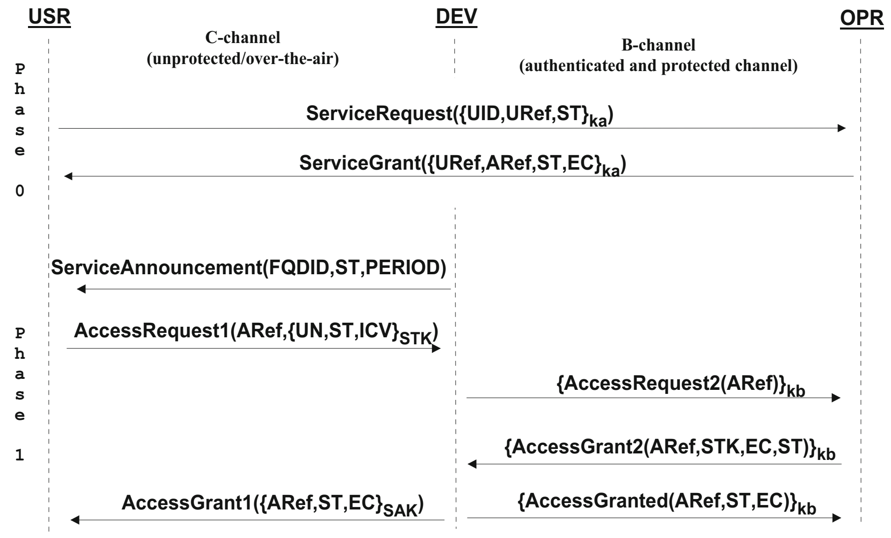
USR
DEV
OPR
C-channel (unprotected/over-the-air)
B-channel (authenticated and protected channel)
Phase 0
ServiceRequest({UID,URef,ST}$_{ka}$)
ServiceGrant({URef,ARef,ST,EC}$_{ka}$)
ServiceAnnouncement(FQDID,ST,PERIOD)
Phase 1
AccessRequest1(ARef,{UN,ST,ICV}$_{STK}$)
{AccessRequest2(ARef)}$_{kb}$
{AccessGrant2(ARef,STK,EC,ST)}$_{kb}$
AccessGrant1({ARef,ST,EC}$_{SAK}$)
{AccessGranted(ARef,ST,EC)}$_{kb}$

Fig. 2. Outline of the PELDA protocol

6.2 Information Elements

In accordance with ECRYPT II [2] recommendations we advocate all keys to be at least 128 bit long. We also mandate that the identities, references and (pseudo-random) nonces be 128 bit long. We will otherwise remain agnostic with respect to the concrete encoding of the identifers.

```
UID   : Globally Unique Permanent privacy-sensitive user identity;
DID   : Globally Unique Permanent public device identity;
SID   : Permanent public server identity;
FQDID : SID||DID; Fully Qualified Device Identity;
URef  : User Reference; Also URef';
ARef  : Access Reference;
UN    : User Nonce;
ST    : "Service Type" identifier;
EC    : "Expiry Condition" identifier;
ICV   : Integrity Check Value;
STK   : Service Type Key;
SAK   : Service Access Key;
TAKx  : Temporary Access Key x; x={i|c}, (integrity|confidentiality)
```

Fig. 3. PELDA phase 1 - Information Elements

6.3 Key Derivation

During PELDA execution three different symmetric keys are derived. These are the “Service Type Key (STK)”, the “Service Access Key (SAK)” and the “Temporary Access Key ($TAKx$)”. The STK and SAK are key deriving keys.

The TAK is used for protection of service content, and it consists of separate integrity ($TAKi$) and confidentiality ($TAKc$) keys. One could also distinguish between uplink and downlink keys, but we have chosen not to do so here.

The Service Type Key (STK) is a key derived for a specific user identified by the “user reference” ($URef$). The key derivation transform uses a symmetric-key encryption function, E, and the bitwise exclusive-or function. The $kdf1$ function has the property that even a party that possess the “input key” and the output key will not be able to deduce the input parameter. The “input key” in question is not an ordinary encryption key, but the service type identifier (ST). The input parameter $URef'$ is a MAC-modified pseudo-random user reference. The $URef$ is required to be confidential to all but the user (USR) and the operator (OPR). The encrypt-and-xor transform it is not new. It was also used in the MILENAGE authentication algorithm set used in cellular systems (UMTS and LTE) [6].

$$kdf1_{ST}(URef') \rightarrow STK \equiv (E_{ST}(URef') \oplus URef') \rightarrow STK \qquad (1)$$

The Service Access Key (SAK) is specific to one device. The key derivation function takes STK as the input key. The input parameters, all non-secret, includes the device identity (DID), the service type (ST), user generated pseudo-random nonce UN and the access reference ($ARef$). The $ARef$ is generated by the operator and must be guaranteed to be unique with respect to the request context (given in the `ServiceRequest` message).

$$kdf2_{STK}(FQDID, ARef, ST, UN) \rightarrow SAK \tag{2}$$

The Temporary Access Key (TAK) is specific to a session or period. The key derivation function takes the SAK as the input key. The only input parameters is the $PERIOD$ identifier (which was broadcast by the device). Whenever the $PERIOD$ indication in the `ServiceAnnouncement` changes the TAK keys must be re-computed. The expiry condition (EC) may not necessarily coincide with the $PERIOD$ announcement, and that expiry of EC may lead to expiry of the whole service context irrespective of $PERIOD$ expiry.

$$kdf3_{SAK}(PERIOD) \rightarrow TAKi||TAKc \tag{3}$$

6.4 PELDA Protocol Description

We now present the PELDA protocol in an augmented Alice-Bob notation. We assume the presence of a pseudo-random number function (prf), suitable (block cipher) symmetric-key primitives and a MAC function. We also assume that keys for the A-channel (ka) are agreed prior to PELDA execution and that keys for the B-channel (kb) are available during PELDA phase 1 execution.

PELDA Phase 0 - Access Agreement

0. `Preparations (pre-computation possible)`
   - `USR:` $prf(\cdot) \rightarrow URef$

1. `USR`$\rightarrow$`OPR: ServiceRequest(`$\{UID, URef, ST, \}_{ka}$`)`
   - `OPR: Decrypt the ServiceRequest message and prepare the response.`
   - `OPR:` $prf(\cdot) \rightarrow ARef$

2. `OPR`$\rightarrow$`USR: ServiceGrant(`$\{URef, ARef, ST, EC\}_{ka}$`)`

PELDA Phase 1 - Initial Registration

0. `DEV`$\rightarrow$`all: ServiceAnnouncement(FQDID,ST,PERIOD)`
   - `USR:` $prf(\cdot) \rightarrow UN$
   - `USR:` $MAC_{ka}(URef, FQDID) \rightarrow URef'$
   - `USR:` $kdf1_{ST}(URef') \rightarrow STK$
   - `USR:` $kdf2_{STK}(FQDID, ARef, ST, UN) \rightarrow SAK$
   - `USR:` $MAC_{STK}(FQDID, PERIOD, ARef, ST, EC) \rightarrow ICV$

1. `USR→DEV: AccessRequest1(ARef,`$\{UN, ST, ICV\}_{STK}$`)`
2. `DEV→OPR: AccessRequest2(`$\{ARef\}_{kb}$`)`
   · `OPR:` $MAC_{ka}(URef, FQDID) \rightarrow URef'$
   · `OPR:` $kdf1_{ST}(URef') \rightarrow STK$

3. `OPR→DEV: AccessGrant2(`$\{ARef, STK, EC, ST\}_{kb}$`)`
   · `DEV: Decrypt` $\{UN, ST, ICV\}_{STK}$ `from AccessRequest1`
   · `DEV: Verify:` $MAC_{STK}(FQDID, PERIOD, ARef, ST, EC) = ICV$
   · `DEV:` $kdf2_{STK}(FQDID, ARef, ST, UN) \rightarrow SAK$

4. `DEV→USR: AccessGrant1(`$\{ARef, ST, EC\}_{SAK}$`)`
   `DEV→OPR: AccessGranted(`$\{ARef, ST, EC\}_{kb}$`)`
   · `USR,DEV:` $kdf3_{SAK}(PERIOD) \rightarrow TAKi||TAKc$

PELDA Phase 2 - Rekeying

Re-keying of $TAKx$ takes place when the $PERIOD$ identifier in the broadcast message `ServiceAnnouncement` changes.

7 Protocol Analysis

7.1 Complexity and Efficiency Aspects

The computational cost of the PELDA protocol is modest and only symmetric-key primitives are used. We foresee no problem in this area and see no need for a more detailed analysis of this aspect.

We note that with a cipher block size of 128 bits, all messages will conformably fit within 4-5 blocks. This makes the communications complexity quite modest and we see no need for a more detailed analysis of this aspect either.

With respect to round-trip delays, a full round-trip from the user, via the device and to the operator is required. The delay should not be prohibitive, and since we have required the operator to be online a full round-trip cannot be avoided. Observe that `AccessGrant1` and `AccessGranted` are sent in parallel and thus do not induce any additional delay. With this in mind, we conclude that the PELDA phase 1 protocol is indeed a lightweight protocol.

7.2 Brief Security Analysis

PELDA Phase 0: We have that the A-channel is authenticated and protected. Thus, we postulate that security is maintained for the A-channel.

PELDA Phase 1: The user initially generates the $URef$ and UN elements, which are pseudo-random and which are unique and unpredictable. The STK is derived by the user and is known to user to be a fresh secret key. The $ARef$ is known to the user to be associated with the fresh $URef$ (phase 0). The user can thus assume that the $ARef$ is fresh and unique.

In `AccessRequest1` the user sends $ARef$ to the device. The device forwards $ARef$ to the server over authenticated and fully protected the B-channel. The device fully trusts the operator. Thus, when the device receives the `AccessGrant2` message it has assurance that the $ARef$ is valid and that it is associated with STK, EC and ST. By the ICV the device also has assurance that access attempt is indeed for it, for the $PERIOD$ (timeliness) and for and for the user ($ARef$). Subsequent to `AccessGrant1` the user has assurance that the device was recognized by the server through the use of SAK. (SAK can only be derived by a party which knows STK and $ARef$). The user already has assurance of STK and $ARef$, and it therefore accepts the device as being valid.

The server does not get explicit assurances of the user during PELDA phase 1, but relies on the device to ascertain the user ($Aref$). However, the server has instructed the device (in `AccessGrant2`) only to provide services as agreed for $ARef$ (encoded in ST, EC), and so it has covered its needs.

7.3 Brief Privacy Analysis

Our main privacy requirements are that the device should not be allowed to know the user identity. Since the device is never actually given the user identity, neither the UID or the URef, we may conclude that the requirement is trivially fulfilled. However, improper use of the $ARef$ could lead to an intruder constructing an emergent identity for the user. Thus, one must assure that the $ARef$ is not exposed (`AccessRequest1`) too many times.

8 Summary and Concluding Remarks

8.1 Summary

We have presented the privacy enhanced lightweight device authentication (PELDA) protocol. The goal of the protocol was to facilitate access to publicly operated IoT-based services such that the user may concentrate of service access rather than on device access.

Privacy is a concern and in particular we have the concern that inexpensive and widely distributed IoT devices may not provide the best protection of privacy sensitive data. The PELDA protocol was designed with this in mind and it will not unduely store privacy sensitive data at the devices. Furthermore, since the PELDA protocol uses a disposable "access references" in place of a permanent identity, there is very little privacy information that the device can leak/divulge. Corollary, subscriber privacy will suffer if $ARef$ is re-used for a prolonged period. The drawback to the PELDA scheme is that it requires the operator to be online, with respect to the device, during the initial device access. This is an unavoidable consequence of the PELDA protocol requirements, but we do not see this as a very limiting restriction in a future with almost pervasive internet connectivity. The bare-bones PELDA protocol is a simple protocol with few roundtrips, a relatively small payload and modest crypto-performance requirements.

8.2 Further Work

We intend to implement variants of the PELDA protocol with an aim to investigate how these protocols cover a wider set of use-cases. We also intend to develop formal models and carry out formal verification of selected aspects. We intend to use the AVISPA (`www.avispa-project.org`) and/or the AVANTSSAR (`www.avantssar.eu`) tools. Here of course we would in particular investigate privacy properties (the formal security modeling tools tend to cater well for entity authentication etc already). Still, we tend to agree with Gollmann [8] in his reluctance to trust formal verification to prove any protocol correct (Gollmann even refers to proofs as a non-goal of formal verification).

Finally, we add that security, privacy and performance comparisons with other alternatives should be conducted. Amongst the alternatives are the "Identity and Access Services" platform from the Kantara Initiative (formerly Liberty Alliance) and solutions based on the US federal initiative "Open Identity Solutions for Open Government" (More information at `http://www.idmanagement.gov`.).

8.3 Concluding Remarks

We have presented the PELDA protocol and we currently believe that it is an adequate protocol for device access. There remains aspects of the protocol that need deeper analysis, but initial investigations seems to indicate that it is secure and that it provides credible user privacy. Furthermore, the simplicity of the protocol indicates that it will perform well and it should scale well too.

References

1. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
2. Smart, N. (ed.): ECRYPT II Yearly Report on Algorithms and Keysizes (2009-2010). Rev.1, 30. March 2010, ECRYPT II (ICT-2007-216676), European Network of Excellence in Cryptology II (2010)
3. Dierks, T., Rescorla, E.: RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, IETF, 08-2008
4. Housley, R., Polk, W., Ford, W., Solo, D.: RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, IETF (April 2002)
5. Køien, G.M.: Entity Authentication and Personal Privacy in Future Cellular Systems. River Publishers, Aalborg (2010)
6. 3GPP, TS 35.205: 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 1: General (Release 9), Sophia Antipolis, France (December 2009)
7. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer (1998)
8. Gollmann, D.: Analysing Security Protocols. In: Abdallah, A.E., Ryan, P.Y.A., Schneider, S. (eds.) FASec 2002. LNCS, vol. 2629, pp. 71–80. Springer, Heidelberg (2003)