# Analysis of Direct Anonymous Attestation using Murphi

Ilya Pirkin, Sudip Regmi

CS 259 Project Report. March 14, 2008

## 1. Introduction

In this project, we modeled and analyzed the security properties of the Direct Anonymous Protocol [1] using Murphi. We simplified the protocol model using security primitives which we considered unbreakable and modeled our abstractions in Murphi. Running simulations in Murphi, we were able to confirm several known attacks in the protocol explained in [2] and [3]. In addition, the model uncovered an issue with cross-site verifications that we will discuss below.

## 2. Protocol Overview and Murphi model

Diagram 1 below shows a simple outline of the DAA protocol. As shown, there are three major agents: the Issuer, Platforms (each consists of the Host and TPM), and Verifiers.

The final goal of the DAA protocol is to provide a way for the verifiers to ensure the authenticity of the platforms without revealing their identity. A platform is considered authentic if it contains an authentic TPM inside and has been correctly authenticated and authorized by the Issuer.

The protocol has two phases: join and sign/verify. The goal of the join protocol is to set up the platform so it can prove its authenticity to verifiers. The join protocol is initiated by the platform. During the join protocol, platforms choose and commit to a secret value *f*, authenticate themselves to the Issuer (using its Endorsement Key *EK*). The Issuer then generates a DAA Certificate from the platforms commitment to *f* and Issuer's short-term key PKI. The platform stores its DAA Certificate internally.

The sign/verify protocol is initiated by a verifier with a platform. During the protocol, the platform generates a message m and signs it using the DAA Certificate received during the join protocol.

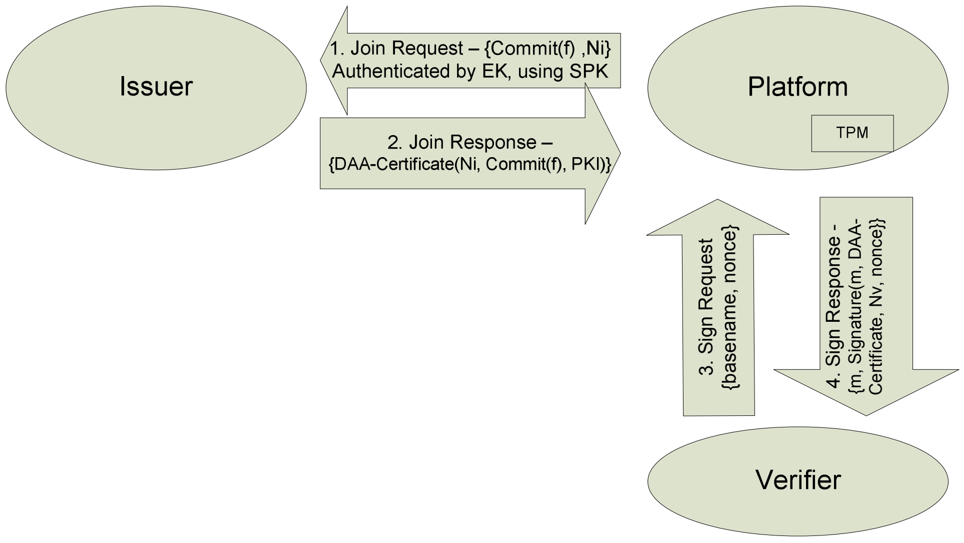
Issuer
1. Join Request – {Commit(f) ,Ni}
Authenticated by EK, using SPK
Platform
TPM
2. Join Response –
{DAA-Certificate(Ni, Commit(f), PKI)}
3. Sign Request
{basename, nonce}
4. Sign Response -
{m, Signature(m, DAA-Certificate, Nv, nonce}}
Verifier

**Diagram 1**

Both in join and sign/verify protocol a pseudonym is constructed from the TPM secret *f* using either Issuer's "basename" (during Join protocol) or Verifier's "basename" (during Sign protocol). The construction of the pseudonym prevents anyone from extracting the secret *f* from it or from linking one pseudonym to another as long as the pseudonyms were generated using different basenames.

We built two models of the DAA protocol: simple and detailed.

- The simple model considers the platform as a whole and does not model interaction and the messages between TPM and the Host. As a consequence the protocol has only 4 different messages and 3 players. Diagram 1, the same as used for the outline, shows its functionality. The level of the primitives in the messages is also very abstract. For example, DAA signature is modeled as a whole – it is modeled as a structure with public key and the TPM secret *f* it, which were used to generate it.
- The detailed model considers the interaction between the TPM and Host. The level of security primitives is close to the primitives used in the original spec.

One of specifics of the DAA protocol is extensive usage of proof of knowledge (PK). For example, after the Platform transmits the commitment to f to Issuer, they run the PK protocol where the Platform proves that it correctly computed the commitment and in fact knows value *f*. As a part of the PK protocol both parties choose random nonces which are used to randomize the PK.

One way to model the PK protocol is in fact create a message with the random initial nonces and then send a separate PK message which would contain the initial PK nonces and the commitment being proven. Since PK is considered a trusted security primitive the intruder would not be able to modify anything in it. The recipient of the PK would compare the nonce and the commitment in the PK with previously received value and would abort the protocol if they don't match (this would model unsuccessful attempt to prove the knowledge).

In our model we chose even simpler implementation of the PK protocols – the PK protocol messages are not modeled at all, but the intruder capabilities are limited so it cannot modify the content of the messages which are protected by PK protocol. The intruder also cannot generate protected messages if it doesn't know the value being proven or replay them since each PK protocol is randomized by exchanging nonce values. The only things which the intruder is capable of are read the values from protected messages and redirect them interactively so the PK protocol can be executed by the parties.

## 3. Security Properties

### *3.1 Correctness*

The verifier completes the protocol for message *m* only if:

- *m* was signed by an honest TPM using the verifier's basename;
- DAA Certificate which was used to generate signature for m was issued by an honest Issuer for the same TPM and its secret *f* before signing message *m*.
- The issuer which TPM joined is the same as the issuer which the verifier obtained from the trusted system.
- TPM is not on the rogue list

The correctness property was modeled by recording certain actual values “behind the scene”, during the protocol execution. In particular, each issuer was recording which platform it authenticated and issued DAA Signature. At the last step of the sign protocol the model used function CheckCorrectness which checked:

- whether the message signature was really singed by an honest platform;
- whether the issuer has in fact authorized the platform.

### *3.2 Anonymity*

A user transaction of an honest platform is anonymous, i.e., it cannot be linked with its Endorsement Key (EK).

This condition was modeled by the Intruder rule which checked certain values in the network messages which appear in the join and sign protocol transcripts. The intruder could also send sign/verify requests to the platform to force it to generate responses that could reveal its identity. If the value is the same in the join response message and in the sign response message originating from the same platform then this value could be used to link the join and sign transactions

The following values in the network messages were tested: Issuer’s short-term public key PKI; platform basename.

### *3.3 Unlinkability*

Transactions of an honest platform with different Verifiers are not linkable with each other.

This condition was modeled by the Intruder rule which compares certain values in the network messages in the join and sign protocol transcripts. In particular, the following conditions were tested:

- Whether a value is the same in two messages of the same type originating from the same platform (value could be used to link the sign/verify transactions);
- Whether this value is different in the messages of the same type originating from different platforms (so this value is not always the same for all sign responses).

The values in the messages which were tested were: Issuer’s short-term public key PKI; platform basename.

## 4. Attacks and other Findings

### 4.1 *Attacks on Anonymity*

Any attack on privacy has to reveal the identity of the platform. As only the Issuer knows the identity of the platform, all attacks on privacy involve a colluding Issuer. So, an attack on privacy involves linking DAA signatures to particular runs of the Join Protocol.

#### 4.1.1 *Rudolph Attack*

[2] describes a simple attack where the Issuer generates a new short-term key for different platforms. At this phase platforms are not anonymous as the Intruder can build a table of Public Key used with corresponding Endorsement Key (EK) of the TPM. Later, when the platform generates a DAA Signature, the Intruder can determine the identity of the platform by looking up the EK by the public key used to generate the signature.
In our model we have an Issuer mode which assigns different PKI to different platform during the join operation. Later the rule which checks anonymity reports an error.

#### 4.1.2 *Basename attack on Anonymity*

In [3], authors describe  potential flaws in the use of basenames that breaks anonymity. When the Issuer and the verifier are the same entity, then it likely that they will use the same "basenames" in the Join as well as the Sign/Verify parts of the DAA. This will result in the same pseudonym being used in both cases. So, this-entity can easily reveal the identity of the TPM during the sign/verify stage by keeping a mapping of pseudonym and the Endorsement key.
In our model we have an Intruder rule which sends sign requests to the platforms using the Issuers' basename. Later the rule which checks anonymity reports an error.

### 4.2 *Attacks on Unlinkability*

Any attack on Anonymity will also break Unlinkability, Such an attack on unlinkability should allow an attacker to link two DAA Signatures produced for two different verifiers and link them both with a particular TPM. [3] describes how the "basenames" could be misused in DAA that can break the unlinkability property. Since there are no guidelines on the basenames, two verifiers could be using the same "basenames". The same "basename" would result in the same pseudonym in the sign/verify transcript.

### 4.3 *Possibility of a cross-site attack*

While working on the modeling of the DAA protocol we encountered a problem of formulating the correctness condition for DAA protocol. In conventional peer-to-peer systems the correctness is usually introduced in the way the actual signer of the message is the signer known to the verifier.

However, in anonymous signing system like DAA, this condition was broken easily with a simple MITM: the Intruder redirects the verifier's request to another honest platform that would sign the message and send it back to the verifier. But this should not create any security problem as long as the platform which responded is still honest and belongs to the same network or service. But consider a situation when a sign request is redirected to a completely different network which also has its own issuers and honest platforms. The DAA protocol does not require that the verifier checks whether the issuer is in fact the right one. If no precautions are taken against this scenario the verifier may end up accepting the signature from a platform from a network or service completely different from the intended. In order to avoid that we had to use the correctness condition specified in section 3 and assumed that such a check is done later by the verifier.

## 5. Fixes

For the attacks described above, there have been suggestions for the fixes to the DAA protocol. In this section, we will describe those fixes to the protocol and detail how it was implemented in the Murphi Model.

### 5.1 *Fix for the Rudolph attack on Anonymity*

There are a number of different ways to prevent the Rudolph attack as described in [5]. One option is to modify the DAA specification and disallow any self-signed certificates by the Issuer. However, this would transfer the burden and trust onto the CA and we need to assume that the CA won't certify arbitrary number of certificates for an Issuer.

Alternatively, we could assume that there exists another trusted Third Party, called the Trusted Auditor as described in [5]' which verifies from time to time that the Issuer is using the same key at least n number of times.. The value n can be increased as required for privacy. In the model, we control this with a flag that control whether or not the issuer generates new self-signed key for each new platform.

### 5.2 *Anonymity - preventing the same pseudonym for Verifier and Issuer.*

The fix for this attack as described in [3]is to include the Issuer/Verifier bit into basename so they always produce different pseudonyms.

### 5.3 *Linkability – Prevent misuse of Basenames*

Although DAA provides user controlled linkability, it is still possible that that the transactions can be linked between two different verifiers that use the same basename without the platform's consent. Since the DAA protocol does not mandate any security requirements for the basename the fix for this issue is elaborate and requires a well-understood mechanism for basename use among the various entities. We don't fix this in our model.

## 6. Conclusion

- Using the simple DAA model (where TPM and Host are modeled as one entity) we were able to reproduce known DAA attacks and model their fixes.
- No new attacks were found (except possible cross-site attack described further). Existing attacks demonstrate misuse of the protocol rather than weakness in its core logic.
- Using the simple DAA model we demonstrated the possibility of a cross-site attack in DAA if the protocols at higher levels do not check if the Issuer resides on the same network as the platforms.
- Detailed DAA model (which models interaction between TPM and Host and the security primitives between them) was too complex to be useful and caused Murphi to run out of memory.
- Correctness condition in anonymous systems is different from that of in peer-to-peer systems. In peer-to-peer system the correctness is held when interacting agents agree on the output of the security protocol. In anonymous system any honest agent may participate in transaction.
- It is difficult to program liveliness condition in Murphi. If the protocol never completes the user will not know it.

## 7. References

1. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington DC, USA, October 25-29, 2004, pages 132-145. ACM Press, 2004.
2. Rudolph, C, 2007, in IFIP International Federation for Information Processing, Volume 232, New Approaches for Security, Privacy and Trust in Complex Environments, eds. Venter, H., ElotT, M., Labuschagne, L., Eloff, J., von Solms, R., (Boston: Springer), pp. 443–448.
3. Smyth, B., Ryan, M. & Chen, L. (2007) Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators. In proceedings of the Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks. Lecture Notes in Computer Science (LNCS), volume 4572, pp. 218-231, Springer-Verlag.
4. Smyth, B. (2006) Direct Anonymous Attestation (DAA): An implementation and security analysis. Master's dissertation, School of Computer Science, University of Birmingham, UK.
5. Adrian Leung, Liqun Chen and Chris J. Mitchell: On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA) in A-R. Sadeghi and P. Lipp (Eds): Trust 2008, Villach, Austria, March 11-12, 2008. Proceedings, volume 4xxx of Lecture Notes in Computer Science, pp. xx. Springer-Verlag, Berlin, Heidelberg, 2008.