Computer Science Project Proposal

Pico without public keys

S. Thang, St Edmund's College

Originator: Dr Graeme Jenkinson

14 October 2015

Project Supervisor: Dr Graeme Jenkinson

Director of Studies: Dr Robert Harle

Project Overseers: Dr J. Crowcroft & Dr T. Sauerwald

Introduction

Pico is an existing research project seeking to replace passwords with physical tokens. In its existing embodiment, the Pico device authenticates the user to remote services using a public key based security protocol called SIGMA-I.

This project will replace SIGMA-I with a symmetric key based alternative. The advantages of this approach include: minimizing changes required for service providers to adopt Pico, lower energy/computing requirements for wearable platforms and future proofing for attacks against public key cryptography[1].

Starting Point

Pico is currently implemented using a browser extension (Pico Lens) and mobile application (Pico App). When the user navigates to a supported site, Pico Lens displays a QR code. The user then expresses the intent to login by scanning the QR code with the Pico App. The Pico App creates an encrypted pipe to the service to authenticate with using the public key protocol SIGMA-I, and proceeds to send the credentials without any further encryption or hashing over the encrypted pipe. Finally, the Pico App delegates the login session to the user's browser via the Pico Lens extension[2], thus completing the authentication process.

Pico Android is built upon JPico, a library written in Java. JPico provides an extensible framework for implementing the cryptographic protocols used by Pico, provided in the form of a Prover interface and its implementations. An implementation of the Prover interface which uses SIGMA-I public key protocol currently exists.

[1] Stajano et al, *Pico without public keys* , `http://www.cl.cam.ac.uk/~fms27/papers/2014-StajanoLomChr-postquantum.pdf`

[2] Jenkinson et al, *Relay attacks on visual code authentication schemes*, `http://www.cl.cam.ac.uk/~fms27/papers/2014-JenkinsonSpeWarETAL-phish.pdf`

Resources Required

For this project I shall use my Windows laptop (with Android SDK and eclipse) and two Android phones for development. Access to the Pico app source code is required, from the projects git repository hosted on git.csx.cam.ac.uk, provided by Dr. Graeme Jenkinson. I accept full responsibility for my machine/phone and I have made contingency plans below to protect myself against hardware and/or software failure.

All code and resources relating to my dissertation will be pushed daily to Github. Each work package will start on a new branch and merged into master on completion. Weekly working copy snapshots will be made to Dropbox. As a copy will be located on (1) my computer, (2) GitHub and (3) Dropbox, there exists multiple redundancies.

Should my computer fail, I shall fall back on the MCS, which is adequate for Java development. Furthermore, if my phones fail, the Pico team has spare phones for testing.

Work to Be Done

The project breaks down into the following sub-projects:

1. Establish a secure pipe between Pico and the service to authenticate with without the use of public key protocols by implementing the Needham-Schroeder Symmetric Key protocol in the JPico library, a well-known symmetric encryption algorithm which forms the basis for Kerberos.

2. Create a corresponding service/endpoint for the client in (1), including the Key Distribution Centre for the symmetric key protocol. Tests should be written for robustness.

3. Perform authentication over the established secure pipe. This would involve implementing the scheme suggested by Bonneau[3] to ensure only hashes of the password and username are sent, to further limit the potential impact of a compromise in database or network security.

4. Create a corresponding service which will implement the authentication described in (3), primarily storing the hashes and authenticating incoming requests.

Success Criterion for the Main Result

The project will be considered a success if:

1. The Pico Android application successfully and reliably creates an encrypted pipe with a corresponding service without the use of public keys.

2. The Pico Android application can perform both registration and authentication without sending the password in clear text over the encrypted pipe.

3. A user can authenticate to a service (specified in the technical work section) with the Pico Android application.

[3]Bonneau, *Getting web authentication right: A best-case protocol for the remaining life of passwords*, `http://www.jbonneau.com/doc/B11-SPW-web_auth_right.pdf`

Further, the extent of the project's success can be further evaluated by one or more of:

1. The power requirements of symmetric key authentication over the existing baseline of public key authentication. There are Android applications to check for application/overall power usage[4].

2. The time taken for authentication, both including and excluding communication overhead, to be evaluated in comparison to the existing public key authentication.

Possible Extensions

If the core project is completed early, I will implement a Wordpress plugin for the project to show the viability of symmetric key authentication on a production platform.

In addition, the protocol can be extended to have a revocation component, such that lost Pico devices can be revoked.

Finally, assertions on the security implications of relying on a third party KDC over the root CA used in public key protocols can be made through BAN logic.

Timetable: Workplan and Milestones to Be Achieved

Planned starting date is 16/10/2015.

1. **Michaelmas weeks 2–3** Learn to use the Android SDK and the Pico codebase. Read papers written by Bonneau and Needham-Schroeder Symmetric Key protocol. Deliver written findings for inclusion in January progress report.

2. **Michaelmas weeks 4–5** Implement the client, server and KDC for the Needham-Schroeder Symmetric Key protocol. Deliver code for supervisor review.

3. **Michaelmas weeks 6–8** Implement the client and server for the authentication scheme suggested by Bonneau. Deliver code for supervisor review, unit/integration tests and results.

4. **Michaelmas vacation** Evaluate the core project's speed and power requirements as described success criterion. Slack time here for polish/catch up. Deliver dissertation draft chapters on core project evaluation.

5. **Lent weeks 0–2** Write progress report and generate corpus of test examples.

6. **Lent weeks 3–5** Implement project extension: Wordpress plugin. Write tests and evaluate real world performance. Deliver code for supervisor review, working demo.

7. **Lent weeks 6–8** Implement project extension: Credentials revocation. Examine impact security and power/speed requirements. Deliver code for supervisor review.

[4]Zhang et al, *Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones*, `http://robertdick.org/publications/zhang10oct.pdf`, with implementation on `https://github.com/msg555/PowerTutor`

8. **Easter vacation:** Writing dissertation main chapters. Slack time included here to polish/finish extensions. If time permits, read up on BAN logic and attempt to evaluate security aspect. Deliver dissertation draft.

9. **Easter weeks 0–2:** Further evaluation and complete dissertation.

10. **Easter week 3:** Proof reading and then an early submission so as to concentrate on examination revision.